ABBA Neural Networks: Coping with Positivity, Expressivity, and Robustness

Ana-Antonia Neacşu, Jean-Christophe Pesquet, Vlad Vasilescu, Corneliu Burileanu

▶ **To cite this version:**

Ana-Antonia Neacşu, Jean-Christophe Pesquet, Vlad Vasilescu, Corneliu Burileanu. ABBA Neural Networks: Coping with Positivity, Expressivity, and Robustness. SIAM Journal on Mathematics of Data Science, In press, 6 (3), ⟨10.1137/23M1589591⟩. ⟨hal-04386260v2⟩

HAL Id: hal-04386260

https://hal.science/hal-04386260v2

Submitted on 26 Jul 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons CC BY-NC 4.0 - Attribution - Non-commercial use - International License

ABBA Neural Networks: Coping with Positivity, Expressivity, and Robustness *

Ana Neacșu †, Jean-Christophe Pesquet ‡, Vlad Vasilescu †, and Corneliu Burileanu †

Abstract. We introduce ABBA networks, a novel class of (almost) non-negative neural networks, which are shown to possess a series of appealing properties. In particular, we demonstrate that these networks are universal approximators while enjoying the advantages of non-negative weighted networks. We derive tight Lipschitz bounds both in the fully connected and convolutional cases. We propose a strategy for designing ABBA nets that are robust against adversarial attacks, by finely controlling the Lipschitz constant of the network during the training phase. We show that our method outperforms other state-of-the-art defenses against adversarial white-box attackers. Experiments are performed on image classification tasks on four benchmark datasets.

Key words. Neural Networks, Robustness, Optimization, Lipschitz, Image Classification

MSC codes. 15A48, 68T05, 65K10

Published paper URL. https://epubs.siam.org/doi/abs/10.1137/23M1589591

1. Introduction.

Deep learning methods based on neural network models have received increasing attention in the scientific community, because of their stunning abilities to solve a variety of complex tasks. These powerful systems excel at learning intricate mappings and, in some cases, even surpass human performance. However, deep architectures usually lack interpretability and they may lead to over-parameterized models [13, 37]. Additionally, their robustness is not well-controlled, leaving them exposed to potential adversarial attacks. For instance, [49] demonstrated that by introducing carefully-crafted, low-magnitude adversarial perturbations, neural classifiers could be easily fooled [17]. A way of overcoming the aforementioned challenges consists in introducing some specific constraints in the neural network design. In this article, we are interested in nonnegativity and stability constraints on the network weights.

It is widely accepted that humans possess the innate ability to decompose complex interactions into discrete, intuitive hierarchical categories before analyzing them [26]. Conceptually, this evolution towards part-based representation in human cognition can be linked to non-negativity restrictions on the network weights [7]. This idea, along with other factors, has sparked interest in neural networks with non-negative weights. These networks have drawn attention for several reasons. Firstly, they align with human understandability, making them more interpretable. Secondly, the non-negativity constraint can act as beneficial regularization, effectively reducing overfitting issues. Moreover, recent studies have demonstrated that it is possible to derive a tight Lipschitz bound for such networks.

*Received by the editors August 30, 2023; accepted for publication (in revised form) December 7, 2023; published electronically July 15, 2024.
Part of this work was supported by the French ANR Research and Teaching Chair in Artificial Intelligence BRIDGEABLE.

†Speech and Dialogue Laboratory, University Politehnica of Bucharest, Romania (ana_antonia.neacsu@upb.ro), (vlad.vasilescu2111@upb.ro), (corneliu.burileanu@upb.ro).

‡Centre de Vision Numérique, Inria, CentraleSupélec, Gif-sur-Yvette, France (jean-christophe.pesquet@centralesupelec.fr).

This Lipschitz constant serves as a valuable metric for quantifying the robustness of the network, enabling us to design networks with enhanced resilience to adversarial perturbations during the training process. Despite their advantages, one significant drawback of networks with non-negative weights is that they might be less expressive than networks with arbitrary signed weights. In [54], it is shown that standard non-negative networks are not universal approximators [54], a limitation that our work overcomes.

Approach. In this work, we are interested in neural networks having non-negative weights, except for the first and last linear layers. This class of networks obviously constitutes an extension of those having all their linear layers non-negative-valued. We focus on a particular subclass of these networks for which the weight matrices have a structure of the form

$$\begin{bmatrix} A & B \\ B & A \end{bmatrix},$$

thus enjoying a number of algebraic properties. The corresponding networks are subsequently called ABBA networks. Note that weight matrices A and B are duplicated in ABBA networks, thus allowing us to limit the number of parameters.

Contributions. This paper makes several key contributions, which are as follows:

- We show that we can put any arbitrary signed network in an ABBA form. This property holds for fully connected as well as for convolutional neural networks.
- Universal approximation theorems are derived for networks featuring non-negatively weighted layers.
- We present a method for effectively controlling the Lipschitz constant of ABBA networks[1]. The resulting training strategy applies to both fully connected and convolutional cases. Precise Lipschitz bounds are typically NP hard to compute for arbitrary signed networks, but our framework allows us to derive such bounds that are easy to compute.
- Numerical experiments conducted on standard image datasets showcase the excellent performance of ABBA networks for small models. Notably, they exhibit substantial improvements in both performance and robustness compared to networks with exclusively non-negative weights. Moreover, we demonstrate that ABBA networks are competitive with robust networks featuring arbitrarily signed weights, trained using state-of-the-art techniques.

Outline. The rest of the paper is organized as follows. Section 2 offers an overview of the related literature, while in Section 3 our main contributions concerning ABBA architectures are introduced, alongside a list of fundamental properties. Section 4 extends our results to the case of convolutional neural networks and two Lipschitz constant expressions are derived. Section 5 describes the training strategy we employ to generate robust models with respect to adversarial perturbations. Section 6 details the results obtained for different image classification tasks, while Section 7 is dedicated to concluding remarks.

2. Related work.

Non-negative neural networks. Inspired by non-negative matrix factorization (NMF) techniques, the work of [7] introduces non-negative restrictions on the weights to create neural networks in which the hidden units correspond to identifiable concepts. [4] showed that Autoencoders (AE) trained under non-negativity constraints are able to derive meaningful representations that

[1]A full PyTorch implementation of our framework will be made available at https://github.com/Vladimirescu/ABBA-Neural-Networks-torch.

unearth the hidden structure of high-dimensional data. Their method showed promising results from both performance and feature interpretation viewpoints on four different classification tasks. [12] presented the first polynomial-time algorithm for Probably Approximately Correct (PAC) learning 1-layer neural networks with positive coefficients. Moreover, ensuring non-negativity has been shown to have a regularization effect, reducing feature overfitting, which is a very common problem, especially for tasks where the available training data is scarce [35]. Neural networks defining convex functions of their inputs [1] also constitute a subclass of networks with non-negative weights.

Link with other networks. From another perspective, the idea of using redundant weights is reminiscent of siamese networks [6]. These architectures are successfully used to handle similarity learning tasks, such as face verification [50], character recognition [23], and object tracking [19]. Siamese networks compute a similarity metric on the representations of the inputs, after applying the same transformation to each one. Apart from the proven efficiency on solving computer vision tasks, they have lately been employed in NLP problems, e.g., computational argumentation. In [14], it is shown that siamese architectures outperform other baselines trained on convincingness datasets.

Robustness. The robustness of neural networks against possible adversarial attacks is a topic that has received increasing attention since nowadays AI-based solutions are ubiquitous [3, 36]. A sizable body of literature on adversarial attacks and different defense strategies have emerged in recent years as a result of the work in [49]), which revealed the alluring susceptibility of neural networks to adversarial perturbations and proposed a box-constrained L-BFGS algorithm for finding adversarial examples. [15] introduced the FGSM attack as a one-step modification of the input image, following the direction of loss maximization, while [24] incorporated this step into an iterative method known as PGD, seen as an improvement over basic FGSM. DeepFool [33] iteratively searches for the closest adversarial point that directs the optimization towards crossing the decision boundary. DDN [42] and FMN [39] attacks fall into the category of projected-gradient methods, using iterative updates of the perturbation vector towards the minimization of its magnitude.

Defensive strategies have been developed to alleviate this robustness issue. [47] divides adversarial defense methods into three categories: adversarial detection, gradient masking, and robust optimization. Adversarial Training (AT) was first introduced by [15] and later improved by [32]. Recent works on AT [48, 55] have successfully analyzed and refined training techniques, however, no theoretical certificates regarding their behavior in the presence of different adversaries have been established yet. Regularization-based methods, such as [30, 46, 59], include additional terms in their objective, steering the learning process in a direction that leads to better generalization. [40] provides robustness certificates for neural networks with one hidden layer, yielding an upper bound of the error in the presence of any adversary (see [11, 16, 41] for more advanced methods. Randomized smoothing [9, 25, 29, 44, 57] certifies the robustness of a classifier around an input point by measuring the most-likely prediction over Gaussian-corrupted versions of the point.

Lipschitz properties of neural networks. As highlighted by [49], the Lipschitz behavior of a neural model is closely correlated with its robustness against adversarial attacks, providing an upper bound on the response given input perturbations. Controlling the Lipschitz behavior of the network thus offers theoretical stability guarantees. However, computing the exact constant, even for small networks is an NP-hard problem [20], and finding a good approximation in a reasonable time is an open challenge. Several solutions have been proposed lately (see for example: [5, 18, 21, 58]). [45] introduced *deel-lip*, a framework to control the Lipschitz constant of each layer individually, while [2] propose GroupSort networks to ensure robustness. [38] proposed a framework for training

fully-connected neural networks using Lipschitz regularized and constrained techniques, proving their effectiveness in the scenario of Gaussian-added perturbation noise. A recent result in [10] showed that in the case of models with non-negative weights a tight Lipschitz bound can be established, making possible the training of neural network models with certified robustness guarantees.

3. ABBA Neural Networks.

3.1. Problem formulation.

In the remainder of this paper, $\|\cdot\|$ will denote the ℓ_2-norm when dealing with a vector, and the spectral norm when dealing with a matrix.
An m-layer feedforward neural network can be described by the following model.

Model 3.1. T is feedforward neural network if there exists $(N_i)_{1\leqslant i\leqslant m} \in (\mathbb{N}\setminus\{0\})^m$ such that

$$T = T_m \circ \cdots \circ T_1 \tag{3.1}$$

where, for every layer index $i \in \{1,\ldots,m\}$, $T_i = R_i(W_i \cdot + b_i)$, $W_i \in \mathbb{R}^{N_i\times N_{i-1}}$ is the weight matrix, $b_i \in \mathbb{R}^{N_i}$ the bias vector, and $R_i\colon \mathbb{R}^{N_i} \to \mathbb{R}^{N_i}$ the activation operator. N_i corresponds to the number of inputs at the i-th layer. Such a layer is convolutive if it corresponds to a weight matrix W_i having some Toeplitz (or block Toeplitz) structure.
We will say that the activation operator R_i is symmetric, if there exists $(c_i, d_i) \in (\mathbb{R}^{N_i})^2$ such that

$$(\forall x \in \mathbb{R}^{N_i}) \quad R_i(x) - d_i = -R_i(-x + c_i). \tag{3.2}$$

In other words, $(c_i, d_i)/2$ is a symmetry center of the graph of R_i.

For example, if R_i is squashing function used in CapsNets [43], it is such that

$$(\forall x \in \mathbb{R}^{N_i}) \quad R_i(x) = \frac{\mu\|x\|}{1+\|x\|^2}x. \tag{3.3}$$

with $\mu = 8/(3\sqrt{3})$. It thus satisfies the symmetry property (3.2) with $c_i = d_i = 0$. In addition, R_i is nonexpansive, i.e. it has a Lipschitz constant equal to 1 [10]. Other examples of symmetric and nonexpansive activation operators are presented in Appendix SM1 [2] .

3.2. ABBA Matrices.

We first define ABBA matrices which will be the main algebraic tool throughout this article.

Definition 3.2. *Let $(N_1, N_2) \in (\mathbb{N}\setminus\{0\})^2$. $\mathcal{A}_{N_1,N_2}$ is the space of ABBA matrices of size $(2N_2)\times(2N_1)$, that is $M \in \mathcal{A}_{N_1,N_2}$ if there exist matrices $A \in \mathbb{R}^{N_2\times N_1}$ and $B \in \mathbb{R}^{N_2\times N_1}$ such that*

$$M = \begin{bmatrix} A & B \\ B & A \end{bmatrix}. \tag{3.4}$$

The sum matrix associated with M is then defined as $\mathfrak{S}(M) = A + B$.

We give some of the most relevant properties of these matrices. In particular, we will see that the ABBA structure is stable under standard matrix operations.

Proposition 3.3. *Let $(N_1, N_2, N_3) \in (\mathbb{N}\setminus\{0\})^3$.*

[2] Appendices with number of the form SMx can be found in the supplementary materials.

(i). *If $M \in \mathcal{A}_{N_2,N_1}$, then its transpose $M^\top \in \mathcal{A}_{N_1,N_2}$ and $\mathfrak{S}(M^\top) = \mathfrak{S}(M)^\top$.*

(ii). *If $(M_1, M_2) \in (\mathcal{A}_{N_2,N_1})^2$, then $M_1 + M_2 \in \mathcal{A}_{N_2,N_1}$ and $\mathfrak{S}(M_1 + M_2) = \mathfrak{S}(M_1) + \mathfrak{S}(M_2)$.*

(iii). *If $M_1 \in \mathcal{A}_{N_2,N_1}$ and $M_2 \in \mathcal{A}_{N_3,N_2}$, then $M_2M_1 \in \mathcal{A}_{N_3,N_1}$ and $\mathfrak{S}(M_2M_1) = \mathfrak{S}(M_2)\mathfrak{S}(M_1)$.*

(iv). *$\mathcal{A}_{N_1,N_1}$ is a ring when equipped with the standard matrix addition and product.*

(v). *If A and B are two square matrices of the same size, the eigenvalues of $\begin{bmatrix} A & B \\ B & A \end{bmatrix}$ are those of $A + B$ and $A - B$.*

(vi). *If A and B are two matrices having the same dimensions, the spectral norm of $\begin{bmatrix} A & B \\ B & A \end{bmatrix}$ is equal to $\max\{\|A + B\|, \|A - B\|\}$.*

(vii). *If $M \in \mathcal{A}_{N_2,N_1}$ has non-negative elements, the spectral norm of M is $\|\mathfrak{S}(M)\|$.*

(viii). *Let $A \in \mathbb{R}^{N_2 \times N_1}$ and $B \in \mathbb{R}^{N_2 \times N_1}$, and let $K = \min\{N_1, N_2\}$. Let $(\lambda_k)_{1 \leqslant k \leqslant K}$ (resp. $(\mu_k)_{1 \leqslant k \leqslant K}$) be the singular values of $A+B$ (resp. $A-B$) and let $\{u_k\}_{1 \leqslant k \leqslant K} / \{v_k\}_{1 \leqslant k \leqslant K}$ (resp $\{t_k\}_{1 \leqslant k \leqslant K} / \{w_k\}_{1 \leqslant k \leqslant K}$) be associated orthonormal families of left/right singular vectors in $\mathbb{R}^{N_2} / \mathbb{R}^{N_1}$.*[3] *Then, the singular values of $\begin{bmatrix} A & B \\ B & A \end{bmatrix}$ are $(\lambda_k, \mu_k)_{1 \leqslant k \leqslant K}$ and associated orthonormal families of left/right singular vectors are*

$$\left\{ \frac{1}{\sqrt{2}} \begin{bmatrix} u_k \\ u_k \end{bmatrix}, \frac{1}{\sqrt{2}} \begin{bmatrix} t_k \\ -t_k \end{bmatrix} \right\}_{1 \leqslant k \leqslant K} \Bigg/ \left\{ \frac{1}{\sqrt{2}} \begin{bmatrix} v_k \\ v_k \end{bmatrix}, \frac{1}{\sqrt{2}} \begin{bmatrix} w_k \\ -w_k \end{bmatrix} \right\}_{1 \leqslant k \leqslant K}.$$

(ix). *If A and B are two matrices having the same dimensions,*

$$\operatorname{rank}\left(\begin{bmatrix} A & B \\ B & A \end{bmatrix} \right) = \operatorname{rank}(A + B) + \operatorname{rank}(A - B). \tag{3.5}$$

(x). *Let f be a function from $\mathbb{R}^{(2N_2)\times(2N_1)}$ to $\mathbb{R}^{(2N_2)\times(2N_1)}$. Assume that either f operates elementwise or it is a spectral function in the sense that there exists a function $\varphi\colon \mathbb{R}_+ \to \mathbb{R}_+$ such that*

$$(\forall M \in \mathbb{R}^{(2N_2)\times(2N_1)}) \quad f(M) = \sum_{k=1}^{2K} \varphi(\tilde{\lambda}_k) \tilde{u}_k \tilde{v}_k^\top \tag{3.6}$$

where $K = \min\{N_1, N_2\}$, $(\tilde{\lambda}_k)_{1 \leqslant k \leqslant 2K}$ are the singular values of M, and $\{\tilde{u}_k\}_{1 \leqslant k \leqslant 2K} / \{\tilde{v}_k\}_{1 \leqslant k \leqslant 2K}$ are associated orthonormal families of left / right singular vectors in $\mathbb{R}^{2N_2} / \mathbb{R}^{2N_1}$. Then f maps any matrix in $\mathcal{A}_{N_2,N_1}$ to a matrix in $\mathcal{A}_{N_2,N_1}$.

(xi). *The best approximation of maximum rank $R < \min\{N_1, N_2\}$ (in the sense of the Frobenius norm) to a matrix in $\mathcal{A}_{N_2,N_1}$ belongs to $\mathcal{A}_{N_2,N_1}$.*

(xii). *The projection onto the spectral ball of center 0 and radius $\rho \in]0, +\infty[$ of an ABBA matrix is an ABBA matrix.*

The proofs of these properties are provided in Appendix SM2.

[3] This means that (SM2.11) and (SM2.12) hold.

3.3. Extension to feedforward networks. We will now extend the previous algebraic concepts by introducing the class of ABBA feedforward neural networks. In the following, the structure of an ABBA fully-connected network will be presented from the perspective of investigating its links with standard networks. Such networks make use of weights that respect the structure of ABBA matrices, except for the first and the last layers. More precisely, the first layer maps the input to a twice-higher dimensional space, while the last layer performs a dimension reduction by a factor of 2.

Definition 3.4. *Let $m \in \mathbb{N} \setminus \{0\}$. $\widetilde{T}$ is an m-layer ABBA network if*

$$\widetilde{T} = (\widetilde{W}_{m+1} \cdot + \widetilde{b}_{m+1})\widetilde{T}_m \cdots \widetilde{T}_1 \widetilde{W}_0 \tag{3.7}$$

with $\widetilde{W}_0 \in \mathbb{R}^{(2N_0)\times N_0}$, $\widetilde{W}_{m+1} \in \mathbb{R}^{N_m \times (2N_m)}$, $\widetilde{b}_{m+1} \in \mathbb{R}^{N_m}$, and

$$(\forall i \in \{1, \ldots, m\}) \quad \widetilde{T}_i = \widetilde{R}_i(\widetilde{W}_i \cdot + \widetilde{b}_i) \tag{3.8}$$

$$\widetilde{R}_i \colon \mathbb{R}^{2N_i} \to \mathbb{R}^{2N_i}, \tag{3.9}$$

$$\widetilde{b}_i \in \mathbb{R}^{2N_i}, \tag{3.10}$$

$$\widetilde{W}_i \in \mathcal{A}_{N_i, N_{i-1}}, \tag{3.11}$$

for given positive integers $(N_i)_{0 \leqslant i \leqslant m}$. $\widetilde{T}$ is an m-layer non-negative ABBA network if it is an m-layer ABBA network as defined above and, for every $i \in \{1, \ldots, m\}$, the elements of $\widetilde{W}_i$ are non-negative.

In the remainder of this paper, $\mathcal{N}_{m,\mathcal{A}}$ will designate the class of m-layer ABBA networks and $\mathcal{N}^+_{m,\mathcal{A}}$ will designate the subclass of m-layer non-negative ABBA networks. This latter subclass will be the main topic of investigation in this work. We will also use the notation $\mathcal{N}^+_{m,\mathcal{A}}(\rho)$ to designate the set of neural networks in $\mathcal{N}^+_{m,\mathcal{A}}$ where all the activation operators operate componentwise using the same function $\rho \colon \mathbb{R} \to \mathbb{R}$.

3.4. Link with standard neural networks. In this section, we show that we can reshape Model 3.1 as a special case of a non-negative ABBA network. At each layer $i \in \{1, \ldots, m\}$ of this model, let $W_i^+ = (W^+_{i,k,\ell})_{1 \leqslant k \leqslant N_i, 1 \leqslant \ell \leqslant N_{i-1}} \in [0, +\infty[^{N_i \times N_{i-1}}$ be the positive part of matrix $W_i = (W_{i,k,\ell})_{1 \leqslant k \leqslant N_i, 1 \leqslant \ell \leqslant N_{i-1}}$, i.e.

$$(\forall k \in \{1, \ldots, N_i\})(\forall \ell \in \{1, \ldots, N_{i-1}\}) \qquad W^+_{i,k,\ell} = \begin{cases} W_{i,k,\ell} & \text{if } W_{i,k,\ell} > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{3.12}$$

Let $W_i^- = W_i^+ - W_i \in [0, +\infty[^{N_i \times N_{i-1}}$ be the negative part of W_i, where all the positive elements of W_i have been discarded. Let us now define a non-negative ABBA neural network by using these quantities.

Definition 3.5. *Let $m \in \mathbb{N} \setminus \{0\}$. Let T be the feedforward neural defined in Model 3.1. $\widetilde{T}$ is a network in $\mathcal{N}^+_{m,\mathcal{A}}$ **associated with** T if it satisfies relations (3.7)-(3.11) with*

$$\widetilde{W}_0 = \begin{bmatrix} I_{N_0} \\ -I_{N_0} \end{bmatrix}, \qquad \widetilde{W}_{m+1} = \frac{1}{2}[I_{N_m} \quad -I_{N_m}], \tag{3.13}$$

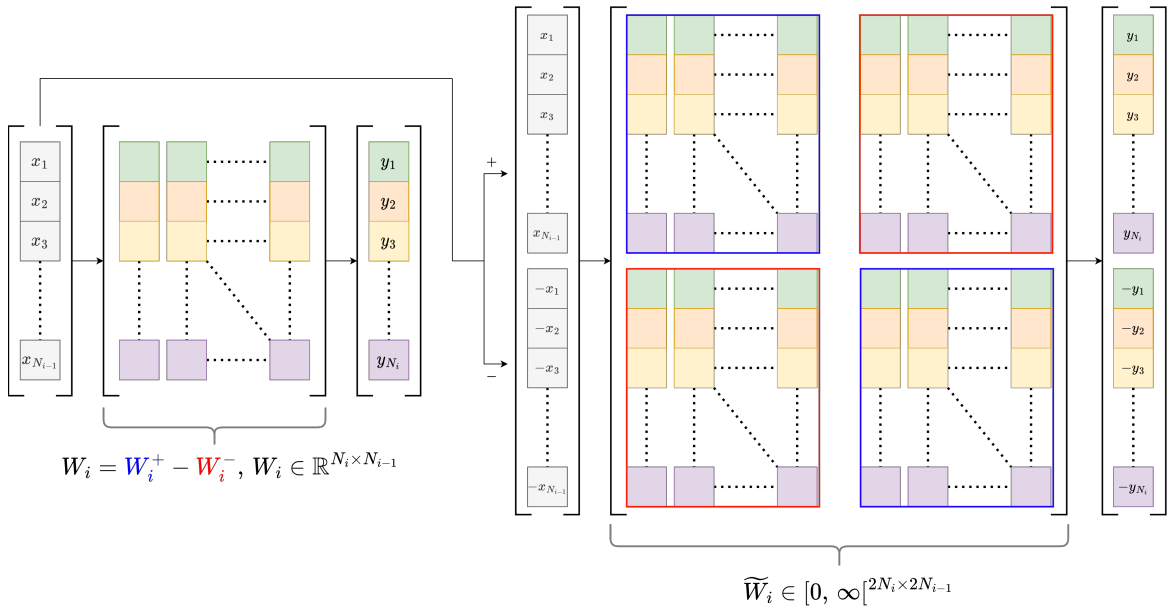


Figure 1: Equivalence between a standard fully-connected layer and its ABBA correspondent.

and

$$(\forall i \in \{1, \dots, m\}) \quad \widetilde{R}_i \colon \begin{bmatrix} x \\ z \end{bmatrix} \mapsto \begin{bmatrix} R_i(x) \\ R_i(z) \end{bmatrix}, \tag{3.14}$$

$$\widetilde{W}_i = \begin{bmatrix} W_i^+ & W_i^- \\ W_i^- & W_i^+ \end{bmatrix}. \tag{3.15}$$

Note that a weight parametrization similar to (3.15) was used in [56] for computing lower and upper bounds on the output of a deep equilibrium layer, but in this article W_i^- has negative values.

As we will show next, the main result is that, if the activation functions are symmetric, network $\tilde{T}$ defined above is identical to network T in terms of input-output relation, for judicious choices of the biases of $\tilde{T}$.

Proposition 3.6. *Let T be the m-layer feedforward network in Model 3.1. Assume that, for every $i \in \{1, \dots, m\}$, the activation operator R_i in the i-th layer of T satisfies the symmetry relation (3.2) where $c_i \in \mathbb{R}^{N_i}$ and $d_i \in \mathbb{R}^{N_i}$. Let $\widetilde{T}$ be the neural network of $\mathcal{N}_{m,\mathcal{A}}^+$ associated with T whose bias vectors $(\widetilde{b}_i)_{1 \leqslant i \leqslant m}$ are linked to those $(b_i)_{1 \leqslant i \leqslant m}$ of T by the relations*

$$(\forall i \in \{1, \dots, m\}) \quad \widetilde{b}_i = \begin{bmatrix} b_i - W_i^- d_{i-1} \\ c_i - b_i - W_i^+ d_{i-1} \end{bmatrix}, \tag{3.16}$$

$$\widetilde{b}_{m+1} = -\frac{d_m}{2}, \tag{3.17}$$

with $d_0 = 0$. Then, for every input, $\widetilde{T}$ delivers the same output as T.

The proof of this proposition is provided in Appendix A. An illustration of the link between fully-connected layers and ABBA matrices is shown in Figure 1.

3.5. Expressivity of non-negative ABBA networks. One of the main advantages of non-negative ABBA networks with respect to standard networks with non-negative weights is that they are universal approximators. More specifically, we have the following result.

Proposition 3.7. *Let $(n_{\mathrm{e}}, n_{\mathrm{r}}) \in (\mathbb{N} \setminus \{0\})^2$. Let $f\colon \mathbb{R}^{n_{\mathrm{e}}} \to \mathbb{R}^{n_{\mathrm{r}}}$ be a continuous function. Let $\mathbb{K}$ be any nonempty compact subset of $\mathbb{R}^{n_{\mathrm{e}}}$ and let $\epsilon \in]0, +\infty[$.*

(i). *Let $\rho\colon \mathbb{R} \to \mathbb{R}$ be a symmetric non polynomial activation function. There exists a network $\widetilde{T} \in \mathcal{N}^{+}_{1,\mathcal{A}}(\rho)$ with $N_0 = n_{\mathrm{e}}$ inputs and $N_2 = n_{\mathrm{r}}$ outputs such that*

$$(\forall x \in \mathbb{K}) \quad \|\widetilde{T}(x) - f(x)\| < \epsilon. \tag{3.18}$$

(ii). *Let $\rho\colon \mathbb{R} \to \mathbb{R}$ be a symmetric continuous activation function that is continuously differentiable around at least one point where its derivative is nonzero. Then there exists $m \geqslant 3$ and $\widetilde{T} \in \mathcal{N}^{+}_{m,\mathcal{A}}(\rho)$ with $N_0 = n_{\mathrm{e}}$ inputs, $N_{m+1} = n_{\mathrm{r}}$ outputs, and $2N_i = 2(n_{\mathrm{e}} + n_{\mathrm{r}} + 2)$ neurons in every layer $i \in \{1, \ldots, m\}$ such that (3.18) holds.*

Proof. Proposition 3.6 shows that non-negative ABBA networks can be as expressive as signed networks. Combining this fact with existing universal approximation results for signed networks (see [28] for (i) and [22] for (ii)) allows us to deduce these results. ∎

(i) addresses the case of shallow wide networks where the number of neurons in the hidden layer can be arbitrarily large, while (ii) corresponds to the case of deep networks having a limited number of neurons per layer. An illustration of these results is provided in Appendix SM7.

3.6. Lipschitz bounds for ABBA fully-connected networks. As mentioned in the previous sections, the robustness of neural networks with respect to adversarial perturbations can be evaluated through their Lipschitz constant. However, most of the existing techniques for computing a tight estimate of the constant have a high computational complexity for deep or wide networks, whereas simpler upper bounds may turn out to be over-pessimistic.

Nevertheless, in the context of non-negative weighted neural networks [10] proved that tight approximations to the Lipschitz constant can be achieved. In the following, we extend this result and show that we can derive a simple expression for the Lipschitz constant, using a separable bound, for non-negative ABBA networks.

Proposition 3.8. *Let $m \in \mathbb{N} \setminus \{0\}$ and let $\widetilde{T} \in \mathcal{N}^{+}_{m,\mathcal{A}}$ be given by (3.7)-(3.11). Assume that, for every $i \in \{1, \ldots, m\}$, $\widetilde{R}_i$ is a nonexpansive operator operating componentwise. A Lipschitz constant of $\widetilde{T}$ is*

$$\theta_m = \|\widetilde{W}_{m+1}\| \; \|\mathfrak{S}(\widetilde{W}_m) \cdots \mathfrak{S}(\widetilde{W}_1)\| \, \|\widetilde{W}_0\|. \tag{3.19}$$

The proof of this result is detailed in Appendix B. Note that this bound expression could be easily extended to other norms based on the results in [10].

A standard separable upper bound for the Lipschitz constant [49] for the ABBA network $\widetilde{T}$ considered in the previous proposition is

$$\overline{\theta}_m = \|\widetilde{W}_{m+1}\| \, \|\widetilde{W}_m\| \cdots \|\widetilde{W}_1\| \, \|\widetilde{W}_0\|. \tag{3.20}$$

According to Proposition 3.3(vii), this bound reads also

$$\overline{\theta}_m = \|\widetilde{W}_{m+1}\| \, \|\mathfrak{S}(\widetilde{W}_m)\| \cdots \|\mathfrak{S}(\widetilde{W}_1)\| \, \|\widetilde{W}_0\|, \tag{3.21}$$

which, by simple norm inequalities, is looser than θ_m.

If T is the feedforward network defined in Model 3.1 and we apply Proposition 3.8 to the associated non-negative ABBA network $\widetilde{T}$ of Definition 3.5. We have

$$\|\widetilde{W}_0\| = \|\widetilde{W}_0^\top \widetilde{W}_0\|^{1/2} = \|2\, I_{N_0}\|^{1/2} = \sqrt{2} \tag{3.22}$$

and

$$\|\widetilde{W}_{m+1}\| = \|\widetilde{W}_{m+1} \widetilde{W}_{m+1}^\top\|^{1/2} = \frac{1}{\sqrt{2}}. \tag{3.23}$$

In turn, for every $i \in \{1, \dots, m\}$,

$$\mathfrak{S}(\widetilde{W}_i) = W_i^+ + W_i^- = |W_i|. \tag{3.24}$$

where $|W_i|$ is the matrix whose elements are the absolute values of W_i. Hence the Lipschitz constant of $\widetilde{T}$ in (3.19) reduces to

$$\theta_m = \||W_m| \dots |W_1|\|. \tag{3.25}$$

It then follows from Proposition 3.6 that θ_m is also a Lipschitz constant of T when using symmetric activation functions. Note that this bound was actually already derived in [10, Proposition 5.12].

4. Convolutional networks.

We will now extend the results presented in Section 3 to convolutional layers.

4.1. ABBA convolutional layers.

For any $i \in \{1, \dots, m\}$, $\mathcal{W}_i$ is a convolutional layer with $\zeta_{i-1} \in \mathbb{N}\backslash\{0\}$ input channels, ζ_i output channels, kernels $(w_{i,q,p})_{1 \leqslant p \leqslant \zeta_{i-1}, 1 \leqslant q \leqslant \zeta_i}$, and stride $s_i \in \mathbb{N}\backslash\{0\}$. The output $(y_q)_{1 \leqslant q \leqslant \zeta_i}$ of this layer (prior applying any activation operation) is linked to its input $(x_p)_{1 \leqslant p \leqslant \zeta_{i-1}}$ by

$$(\forall q \in \{1, \dots, \zeta_i\}) \quad \begin{aligned} u_q &= \sum_{p=1}^{\zeta_{i-1}} w_{i,q,p} * x_p \\ y_q &= (u_q)_{\downarrow_{s_i}}. \end{aligned} \tag{4.1}$$

Hereabove, for every $p \in \{1, \dots, \zeta_{i-1}\}$, $x_p = \big(x_p(\mathbf{n})\big)_{\mathbf{n} \in \mathbb{Z}^d}$ designates a d-dimensional discrete signal. Dimension $d = 1$ corresponds to 1D signals and $d = 2$ to images. A similar notation is used for other signals, in particular u_q and $w_{i,q,p}$ with $q \in \{1, \dots, \zeta_i\}$. The d-dimensional discrete convolution is denoted by $*$ and $(\cdot)\downarrow_{s_i}$ is the decimation (or subsampling) by a factor s_i.

The ABBA convolutional layer $\widetilde{W}_i$ associated with $\mathcal{W}_i$ has twice the number of input channels and twice the number of output ones. More specifically, its input consists of ζ_{i-1} signals $(\widetilde{x}_p^+)_{1 \leqslant p \leqslant \zeta_{i-1}}$ and ζ_{i-1} signals $(\widetilde{x}_p^-)_{1 \leqslant p \leqslant \zeta_{i-1}}$. Similarly, its output consists of ζ_i signals $(\widetilde{y}_q^+)_{1 \leqslant q \leqslant \zeta_i}$ and ζ_i signals

$(\widetilde{y}_q^-)_{1\leqslant q\leqslant \zeta_i}$. To make the input-output relations more explicit, let us define the kernels $w_{i,q,p}^+$ and $w_{i,q,p}^-$ analogously to the fully connected case:

$$(\forall \mathbf{n}\in\mathbb{Z}^d)\quad w_{i,q,p}^+(\mathbf{n}) = \begin{cases} w_{i,p,q}(\mathbf{n}) & \text{if } w_{i,p,q}(\mathbf{n})>0\\ 0 & \text{otherwise,}\end{cases}$$
$$w_{i,q,p}^-(\mathbf{n}) = w_{i,q,p}^+(\mathbf{n}) - w_{i,p,q}(\mathbf{n}). \tag{4.2}$$

Then the outputs of the ABBA layer are linked to its inputs by the following relations

$$\begin{aligned}(\forall q\in\{1,\ldots,\zeta_i\})\quad \widetilde{u}_q^+ &= \sum_{p=1}^{\zeta_{i-1}} w_{i,q,p}^+ * \widetilde{x}_p^+ + \sum_{p=1}^{\zeta_{i-1}} w_{i,q,p}^- * \widetilde{x}_p^-\\ \widetilde{u}_q^- &= \sum_{p=1}^{\zeta_{i-1}} w_{i,q,p}^- * \widetilde{x}_p^+ + \sum_{p=1}^{\zeta_{i-1}} w_{i,q,p}^+ * \widetilde{x}_p^-\\ \widetilde{y}_q^+ &= (\widetilde{u}_q^+)_{\downarrow_{s_i}}\\ \widetilde{y}_q^- &= (\widetilde{u}_q^-)_{\downarrow_{s_i}}.\end{aligned} \tag{4.3}$$

The above equations provide the general form of a convolutional ABBA layer when relaxing (4.2).

An alternative formulation of convolutional layers in a matrix form, along with its correspondent d-dimensional spectral representation, is possible (see Appendix SM3). This basically amounts to characterize layer (4.1) by the following matrices

$$(\forall \mathbf{n}\in\mathbb{Z}^d)\quad \boldsymbol{W}_i(\mathbf{n}) = \begin{bmatrix} w_{i,1,1}(\mathbf{n}) & \ldots & w_{i,1,\zeta_{i-1}}(\mathbf{n})\\ \vdots & & \vdots\\ w_{i,\zeta_i,1}(\mathbf{n}) & \ldots & w_{i,\zeta_i,\zeta_{i-1}}(\mathbf{n})\end{bmatrix} \in \mathbb{R}^{\zeta_i\times\zeta_{i-1}}, \tag{4.4}$$

defining the so-called MIMO impulse response of $\mathcal{W}_i$, which plays a prominent role in dynamical system theory [52]. The MIMO impulse response of the ABBA layer $\widetilde{\mathcal{W}}_i$ is then characterized by ABBA matrices:

$$(\forall \mathbf{n}\in\mathbb{Z}^d)\quad \widetilde{\boldsymbol{W}}_i(\mathbf{n}) = \begin{bmatrix} \boldsymbol{W}_i^+(\mathbf{n}) & \boldsymbol{W}_i^-(\mathbf{n})\\ \boldsymbol{W}_i^-(\mathbf{n}) & \boldsymbol{W}_i^+(\mathbf{n})\end{bmatrix} \in [0,+\infty[^{(2\zeta_i)\times(2\zeta_{i-1})}, \tag{4.5}$$

where $\boldsymbol{W}_i^+(\mathbf{n}) = (w_{i,q,p}^+(\mathbf{n}))_{1\leqslant q\leqslant\zeta_i,1\leqslant p\leqslant\zeta_{i-1}} \in [0,+\infty[^{\zeta_i\times\zeta_{i-1}}$ and $\boldsymbol{W}_i^-(\mathbf{n}) = (w_{i,q,p}^-(\mathbf{n}))_{1\leqslant q\leqslant\zeta_i,1\leqslant p\leqslant\zeta_{i-1}} \in [0,+\infty[^{\zeta_i\times\zeta_{i-1}}$. This alternative view will be useful in the following sections.

4.2. Lipschitz bounds for convolutional networks. In this section, we establish bounds on the Lipschitz constant of an m-layer convolutional neural network T. Each linear operator $\mathcal{W}_i$ corresponding to layer $i\in\{1,\ldots,m\}$ will be defined by (4.1). We also define a variable

$$\sigma_i = \prod_{l=1}^{i} s_l \tag{4.6}$$

aggregating strides from layer 1 to layer i. Subsequently, we will assume that, for every $i\in\{1,\ldots,m\}$, the activation operators $(R_i)_{1\leqslant i\leqslant m}$ are nonexpansive operators. Moreover, these operators are applied

componentwise (see Appendix SM1). This means that, for every $i \in \{1,\dots,m-1\}$, there exists a function ρ_i from $\mathbb{R}$ to $\mathbb{R}$ such that

$$\begin{aligned} & (\forall \mathbf{x} \in \mathcal{H}_i) \quad \mathbf{y} = R_i(\mathbf{x}) \\ \Leftrightarrow \quad & (\forall p \in \{1,\dots,c_i\})(\forall \mathbf{n} \in \mathbb{Z}^d) \quad y_p(\mathbf{n}) = \rho_i\big(x_p(\mathbf{n})\big). \end{aligned} \tag{4.7}$$

In Appendix SM4, we derive frequency-based expressions allowing us to calculate bounds on the Lipschitz constant of T. For accurate numerical evaluations, the frequency transform in these expressions has to be replaced by a Discrete Fourier Transform involving a significant number of frequency bins (e.g., 128^d). Due to this fact, a computation bottleneck occurs when MIMO filters are characterized by a large number of input/output channels (e.g., for 2D applications). In the following, we provide an alternative lower-complexity formulation for computing bounds of the Lipschitz constant. In the case of non-negative kernels, we show that this bound is tight.

Theorem 4.1. *Let $(\sigma_i)_{1\leqslant i\leqslant m}$ be the aggregated stride factors of network T, as defined by (4.6), and let*

$$\boldsymbol{W} = (\boldsymbol{W}_m)_{\uparrow \sigma_{m-1}} * \cdots * (\boldsymbol{W}_2)_{\uparrow \sigma_1} * \boldsymbol{W}_1 \tag{4.8}$$

where $(\boldsymbol{W}_i)_{1\leqslant i\leqslant m}$ are the MIMO impulse responses of each layer of network T and, for every $i \in \{2,\dots,m\}$, $(\boldsymbol{W}_i)_{\uparrow \sigma_{i-1}}$ is the interpolated sequence by a factor σ_{i-1} of $\boldsymbol{W}_i$ (see (SM3.9)). For every $\mathbf{j} \in \mathbb{S}(\sigma_m) = \{0,\dots,\sigma_m-1\}^d$, we define the following matrix:

$$\overline{\boldsymbol{W}}^{(\mathbf{j})} = \sum_{\mathbf{n}\in\mathbb{Z}^d} \boldsymbol{W}(\sigma_m \mathbf{n} + \mathbf{j}) \in [0,+\infty[^{\zeta_m \times \zeta_0}. \tag{4.9}$$

Then

$$\theta_m = \Big\| \sum_{\mathbf{j}\in\mathbb{S}(\sigma_m)} \overline{\boldsymbol{W}}^{(\mathbf{j})} \big(\overline{\boldsymbol{W}}^{(\mathbf{j})}\big)^\top \Big\|^{1/2} \tag{4.10}$$

is a lower bound on the Lipschitz constant estimate of network T. In addition, if for every $i \in \{1,\dots,m\}$, $p \in \{1,\dots,\zeta_{i-1}\}$, and $q \in \{1,\dots,\zeta_i\}$, $w_{i,q,p} = (w_{i,q,p}(\mathbf{n}))_{\mathbf{n}\in\mathbb{Z}^d}$ is a non-negative kernel, then θ_m is a Lipschitz constant of T.

The proof of Theorem 4.1 is given in Appendix C.

The constant θ_m in (D.4) is actually equal to the one calculated in Appendix SM4. The following majorization if thus obtained (see (SM4.4)):

$$\theta_m \leqslant \overline{\theta}_m = \|\mathcal{W}_m\| \cdots \|\mathcal{W}_1\|. \tag{4.11}$$

By applying Theorem 4.1 to each individual layer $(\mathcal{W}_i)_{1\leqslant i\leqslant m}$ assumed to be with non-negative kernels, we get the following expression for the upper-bound:

$$\overline{\theta}_m = \prod_{i=1}^{m} \Big\| \sum_{\mathbf{j}\in\mathbb{S}(s_i)} \overline{\boldsymbol{W}}_i^{(\mathbf{j})} \big(\overline{\boldsymbol{W}}_i^{(\mathbf{j})}\big)^\top \Big\|^{1/2}, \tag{4.12}$$

where

$$(\forall i \in \{1,\dots,m\})(\forall \mathbf{j} \in \mathbb{S}(s_i)) \qquad \overline{\boldsymbol{W}}_i^{(\mathbf{j})} = \sum_{\mathbf{n}\in\mathbb{Z}^d} \boldsymbol{W}_i(s_i\mathbf{n}+\mathbf{j}). \tag{4.13}$$

The bound $\overline{\theta}_m$ is generally more tractable than θ_m since it separates the influence of each layer and does not require to compute the global matrix sequence $\boldsymbol{W}$ as expressed by (4.8). However, such separable bounds are usually loose. According to our observations, it turns out that, in the special case of convolutional layers with non-negative kernels, θ_m and $\overline{\theta}_m$ are quite close (see numerical tests in Appendix SM5).

To illustrate these results, the computation of the Lipschitz bound of a layer corresponding to an average pooling is presented as an example in Appendix SM6.

4.3. Bounds for ABBA convolutional networks. Let us extend the previous results to the ABBA context. The linear operators of the considered ABBA network $\widetilde{T}$ are denoted by $(\widetilde{\mathcal{W}}_i)_{0\leqslant i\leqslant m+1}$. The weights in $\widetilde{\mathcal{W}}_0$ and $\widetilde{\mathcal{W}}_{m+1}$ are signed, whereas $(\widetilde{\mathcal{W}}_i)_{1\leqslant i\leqslant m}$ are convolutional layers with d-dimensional non-negative kernels. More precisely, we assume that, for every $i \in \{1,\dots,m\}$, the i-th layer of the ABBA network has $2\zeta_{i-1}$ input channels, $2\zeta_i$ output channels, and stride $s_i \in \mathbb{N}\setminus\{0\}$. The MIMO impulse response of such a layer is of the form (4.5). We make the same assumptions of nonexpansiveness and separability for the activation operators as in the previous section. We recall that $(\boldsymbol{W})_{\uparrow\sigma}$ denotes the interpolated version by a factor σ of a MIMO impulse response $\boldsymbol{W} = \big(\boldsymbol{W}(\mathbf{n})\big)_{\mathbf{n}\in\mathbb{Z}^d}$.

The following result is then established in Appendix D :

Theorem 4.2. *Under the above assumptions on the convolutional ABBA network $\widetilde{T}$, let*

$$(\forall i \in \{1,\dots,m\})(\forall \mathbf{j} \in \mathbb{S}(s_i)) \quad \Omega_i^{(\mathbf{j})} = \sum_{\mathbf{n}\in\mathbb{Z}^d} \mathfrak{S}(\widetilde{\boldsymbol{W}}_i(s_i\mathbf{n}+\mathbf{j})) \in [0,+\infty[^{\zeta_i\times\zeta_{i-1}}, \tag{4.14}$$

where $(\widetilde{\boldsymbol{W}}_i(\mathbf{n}))_{n\in\mathbb{Z}}$ is the MIMO impulse response of the ABBA layer of index i. Then a Lipschitz constant of $\widetilde{T}$ is

$$\overline{\theta}_m = \|\widetilde{W}_{m+1}\| \Big(\prod_{i=1}^{m} \Big\| \sum_{\mathbf{j}\in\mathbb{S}(s_i)} \Omega_i^{(\mathbf{j})} \big(\Omega_i^{(\mathbf{j})}\big)^\top \Big\|\Big)^{1/2} \|\widetilde{W}_0\|, \tag{4.15}$$

where $\|\widetilde{W}_{m+1}\|$ (resp. $\|\widetilde{W}_0\|$) is the spectral norm of the linear operator employed in the last (resp. first layer).

The bound (4.15) will be subsequently used to control the Lipschitz constant of non-negative ABBA networks during their training.

5. Lipschitz-constrained training. The theoretical bounds established in the previous sections provide a relatively easy way of computing a tight estimate of the global Lipschitz constant. We propose a simple approach to control it during the training phase. Since our networks contain mostly layers having non-negative weights and a few layers having arbitrary-signed weights, their Lipschitz constant will be controlled separately and different constraint sets will be handled for each case.

To train a robust ABBA network, we employ a projected version of the well-known ADAM optimizer. Each layer i is parameterized by a vector Ψ_i. In the case of a dense layer, Ψ_i is a vector gathering the elements of weight matrix $\widetilde{W}_i$, the components of the associated bias $\widetilde{b}_i$, and a possible

additional parameter that will be introduced hereafter. For an ABBA layer, Ψ_i is thus a vector of dimension $2N_i(N_{i-1}+1)$ or $2N_i(N_{i-1}+1)+1$. In the context of a 2D convolutional layer, an array w_i of scalar convolutional kernels is substituted for the weight matrix. In the ABBA case, we have $2\zeta_i\zeta_{i-1}$ such kernels. To ensure nonnegativity (if needed) and Lipschitz bound conditions on the weight operator, we project Ψ_i onto a suitable closed and convex constraint set. Considering pairs $(z_k)_{1\leqslant k\leqslant K}$ of inputs images and their associated labels, the operations performed at each epoch $n>0$ to minimize a loss function ℓ are presented in Algorithm 5.1. After each iteration t of the optimizer, we perform a projection $\mathrm{proj}_{\mathcal{S}_{i,t}}$ onto a constraint set $\mathcal{S}_{i,t}$. The definition of this set and the corresponding method for managing the projection is detailed in the following, according to the network type.

Handling Lipschitz constants for fully-connected layers. Consider the network defined by Model (3.5). In the case of fully connected networks, the Lipschitz constant is given by Proposition 3.8, which basically splits the bound into three terms: the first and the last account for the starting and ending layers, respectively, while the middle one encompasses all the ABBA layers. For the two former arbitrary-signed layers, we control the Lipschitz constants individually during training, by imposing a bound on each weight matrix spectral norm. This defines the following two constraints:

$$(\forall i\in\{0,m+1\}) \quad \|\widetilde{W}_i\|\leqslant\overline{\theta}_{m,i}, \tag{5.1}$$

where $\overline{\theta}_{m,i}$ is the imposed Lipschitz bound for the i-th layer. To deal with this constraint, we decompose the weight matrix as $\widetilde{W}_i=\overline{\theta}_{m,i}\widetilde{W}_i'$, which yields the constraint set

$$(\forall i\in\{0,m+1\}) \quad \mathcal{S}_{i,t}=\{\widetilde{W}_i' \mid \|\widetilde{W}_i'\|\leqslant 1\}. \tag{5.2}$$

The projection onto $\mathcal{S}_{i,t}$ is performed by clipping the singular values of $\widetilde{W}_i'$ to 1.

In our proposed training procedure, we set $\overline{\theta}_{m,0}\overline{\theta}_{m,m+1}=1$. This gives the network one degree of freedom to automatically adapt the value of the Lipschiz constant of these two layers. To do so, we adopt the following parametrization

$$\overline{\theta}_{m,0}=\exp(\alpha), \qquad \overline{\theta}_{m,m+1}=\exp(-\alpha), \tag{5.3}$$

where $\alpha\in\mathbb{R}$ is a trainable parameter. It constitutes an extra component of the vector Ψ_i when $i\in\{0,m+1\}$.

In the case of ABBA dense layers, we need to handle two requirements: ensure that, for every $i\in\{1,\dots,m\}$, $\widetilde{W}_i$ is a non-negative ABBA matrix, and to constrain the product of all the weight matrices to be such that $\|\widetilde{W}_m\cdots\widetilde{W}_1\|\leqslant\overline{\theta}_m$. Since $\overline{\theta}_{m,0}\overline{\theta}_{m,m+1}=1$, $\overline{\theta}_m$ corresponds to the target Lipschitz bound for the ABBA network.

For every $i\in\{1,\dots,m\}$, $\widetilde{W}_i$ is parameterized by W_i^+ and W_i^-. We define the following two constraint sets:

$$\mathcal{D}_i=\{(W_i^+,W_i^-)\in(\mathbb{R}^{N_i\times N_{i-1}})^2 \mid W_i^+\geqslant 0 \text{ and } W_i^-\geqslant 0\}, \tag{5.4}$$

$$\mathcal{C}_{i,t}=\left\{(W_i^+,W_i^-)\in(\mathbb{R}^{N_i\times N_{i-1}})^2 \mid \left\|A_{i,t}\begin{bmatrix}W_i^+ & W_i^-\\ W_i^- & W_i^+\end{bmatrix}B_{i,t}\right\|\leqslant\overline{\theta}_m\right\}. \tag{5.5}$$

Here-above, matrix $A_{i,t}$ (resp. $B_{i,t}$) is an ABBA matrix, which is the product of the weight matrices for the posterior (resp. previous layers). In this case, $\mathcal{S}_{i,t}=\mathcal{D}_i\cap\mathcal{C}_{i,t}$. To perform the projection onto the intersection of these two sets, we use an instance of the proximal algorithm presented in [35],

which alternates between elementary projections onto $\mathcal{D}_i$ and projections onto the spectral ball with center 0 and radius $\overline{\theta}_m$. Because of Proposition 3.3(xii), the latter projection allows us to keep the structure of ABBA matrices.

Handling Lipschitz constants for convolutional layers. In the case of convolutional ABBA networks, we derived the bound in (4.15) which consists of the product of $m+2$ terms. The Lipschitz bound constraint is managed by introducing auxiliary variables $(\overline{\theta}_{m,i})_{0\leqslant i\leqslant m+1}$ defining upper bounds for each layer. At iteration t of the algorithm estimates $(\overline{\theta}_{m,i,t})_{0\leqslant i\leqslant m+1}$ of the auxiliary bounds are updated. Similarly to the fully connected case, we use two different types of constraints.
For the i-th ABBA convolutional layer with $i \in \{1,\ldots,m\}$, we consider the constraint set

$$\mathcal{C}_{i,t} = \{\mathcal{W}_i \mid \Big\| \sum_{\mathbf{j}\in\mathbb{S}(s_i)} \Omega_i^{(\mathbf{j})} \big(\Omega_i^{(\mathbf{j})}\big)^\top \Big\| \leqslant \overline{\theta}_{m,i,t}^2\} \tag{5.6}$$

where matrices $(\Omega_i^{(\mathbf{j})})_{\mathbf{j}\in\mathbb{S}(s_i)}$ are linked to the convolution kernels by the linear relation (4.14). By concatenating all these s_i^d matrices horizontally, we obtain a rectangular matrix Ω_i which allows us to reexpress (5.6) in the simpler form:

$$\mathcal{C}_{i,t} = \{\mathcal{W}_i \mid \|\Omega_i\| \leqslant \overline{\theta}_{m,i,t}\}. \tag{5.7}$$

We also have to impose the non-negativity of the filters alongside the stability bound. This corresponds to a constraint set $\mathcal{D}_i$. Projecting onto $\mathcal{S}_{i,t} = \mathcal{C}_{i,t} \cap \mathcal{D}_i$ is performed by using the same iterative proximal algorithm as previously.

For the first and the last layers, we impose similarly that $\|\mathcal{W}_i\| \leqslant \overline{\theta}_{m,i,t}$ with $i \in \{0, m+1\}$. Since the kernels are signed, we resort a frequency formulation (see (SM4.18)) to estimate the spectral norm of the convolutional operator. The procedure we use is described in Appendix SM8.

Convolutional layers are usually succeeded by an ABBA fully connected network. This part will be handled as explained previously. However, we need to set the upper bounds $(\overline{\theta}_{m,i,t})_{0\leqslant i\leqslant m+1}$ used in the convolutional part and the upper bound of the ABBA fully connected part. With a slight abuse of notation, let us denote this latter bound by $\overline{\theta}_{m,m+2,t}$, while the target Lipschitz constant for the global network is still denoted by $\overline{\theta}_m$. We have to deal with the following constraint:

$$\prod_{i=0}^{m+2} \overline{\theta}_{m,i,t} = \overline{\theta}_m. \tag{5.8}$$

We proceed by computing the Lipschitz constants $(\widetilde{\theta}_{m,i,t})_{0\leqslant i\leqslant m+2}$ of the layers after the ADAM update. Then, we set

$$(\forall i \in \{0,\ldots,m+2\}) \quad \overline{\theta}_{m,i,t} = \widetilde{\theta}_{m,i,t} \left(\frac{\overline{\theta}_m}{\prod_{i'=0}^{m+2} \widetilde{\theta}_{m,i',t}} \right)^{\frac{1}{m+3}}, \tag{5.9}$$

which guarantees that (5.8) holds. Update (5.9) can be interpreted as the orthogonal projection onto the constraint set defined by (5.8) after a logarithmic transform of the auxiliary variables. The benefit of such a transform is to convexify the constraint.

Algorithm 5.1 Projected ADAM Algorithm

$\Psi_{i,t}$ – weights of layer i at iteration t
$(z_k)_{1\leqslant k\leqslant K}$ – set of input image-label pairs
$\beta_1,\ \beta_2$ – ADAM hyperparameters

Partition $\{1,\dots,K\}$ into mini-batches $(\mathbb{M}_{q,n})_{1\leqslant q\leqslant Q}$
$t=(n-1)Q+q$
\# sweep mini-batches
for $q\in\{1,\dots,Q\}$ **do**
  for layer i **do**
    $g_{i,t}=\sum_{k\in\mathbb{M}_{q,n}}\nabla_i\ell\big(z_k,(\Psi_{i,t})_{1\leqslant i\leqslant m}\big)$
    $\mu_{i,t}=\beta_1\mu_{i,t-1}+(1-\beta_1)g_{i,t}$
    $\nu_{i,t}=\beta_2\nu_{i,t-1}+(1-\beta_2)g_{i,t}^2$
    $\gamma_t=\gamma\sqrt{1-\beta_2^t}/(1-\beta_1^t)$
    $\widetilde{\Psi}_{i,t}=\Psi_{i,t}-\gamma_t\mu_{i,t}/(\sqrt{\nu_{i,t}}+\epsilon)$
  end for
  for layer i **do**
    $\Psi_{i,t+1}=\mathrm{proj}_{\mathcal{S}_{i,t}}(\widetilde{\Psi}_{i,t})$
  end for
end for

Dataset	Network	Architecture	Accuracy [%]
MNIST	ABBA	Dense	98.33
		Conv	98.70
	Non-Negative	Dense	94.95
		Conv	93.27
	Baseline	Dense	98.35
		Conv	98.68
FMNIST	ABBA	Dense	90.02
		Conv	90.17
	Non-Negative	Dense	84.56
		Conv	83.09
	Baseline	Dense	90.00
		Conv	90.20
RPS	ABBA	Conv	99.08
	Non-Negative	Conv	67.30
	Baseline	Conv	98.86
CelebA	ABBA	Conv	90.21
	Non-Negative	Conv	61.04
	Baseline	Conv	90.17

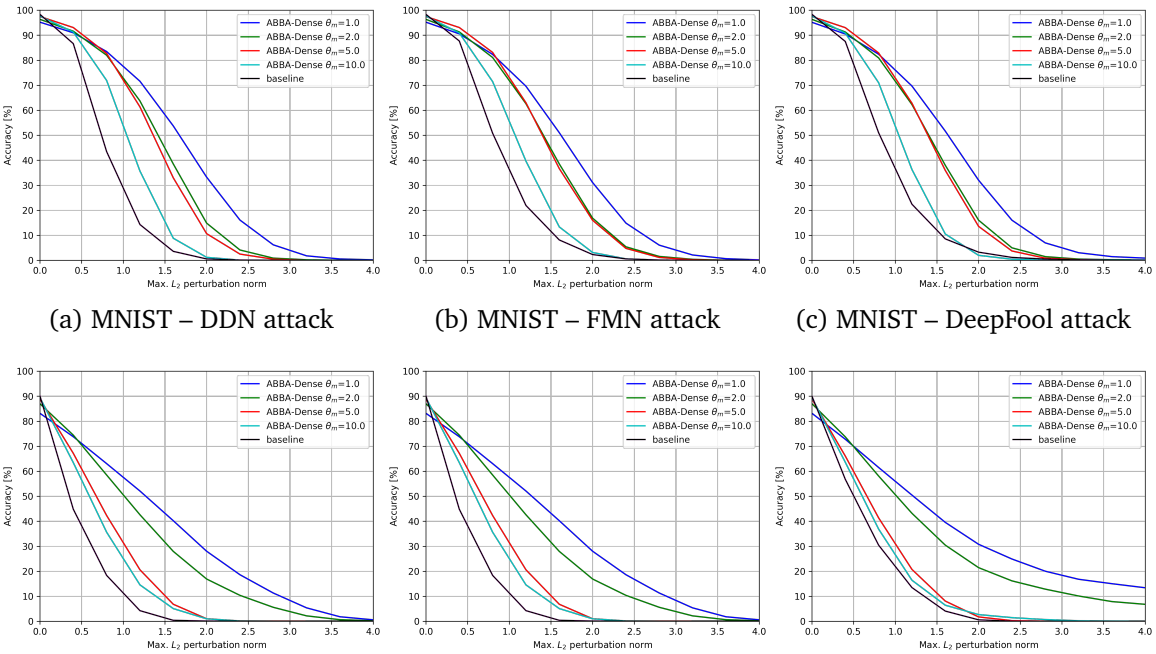
Table 1: Comparison between ABBA, full non-negative and arbitrary-signed (baseline) networks.

6. Experiments. In this section, we show the versatility of ABBA neural networks in solving classification tasks. The objective of our experiments is three-fold:

(i). First, we compare nonnegative ABBA structures with their classic non-negative counterparts and check that our method yields significantly better results in all considered cases.
(ii). We then train ABBA models constrained to different Lipschitz bound values and evaluate their robustness against several adversarial attacks.
(iii). Finally, we compare our proposed approach with three other well-established defense strategies, namely *Adversarial Training* (AT), *Trade-off-inspired adversarial defense* (TRADES) [59], *Deel-Lip* proposed by [45], and *orthonormalization* [2].

We validate our ABBA networks on four benchmark image classification datasets: MNIST, its more complex variant Fashion MNIST [4], a variant [5] of the Rock-Paper-Scissors (RPS) dataset [34], and a binary classification on CelebA [31]. For the last dataset, inspired from [45] where eyeglass detection is performed, we specialized our models on a different attribute, namely to identify whether a person is bald or not. To explore the features of different ABBA network topologies, we experiment with two main types of ABBA architectures: one having only fully-connected layers, further referred to as *ABBA Dense*, and another one which includes a convolutional part for feature extraction, followed by a fully connected classification module, *ABBA Conv*. Depending on the dataset, the particularities of each architecture are slightly different. A detailed description of all the small networks employed in this

[4] https://github.com/zalandoresearch/fashion-mnist
[5] https://github.com/DrGFreeman/rps-cv



(a) MNIST – DDN attack (b) MNIST – FMN attack (c) MNIST – DeepFool attack

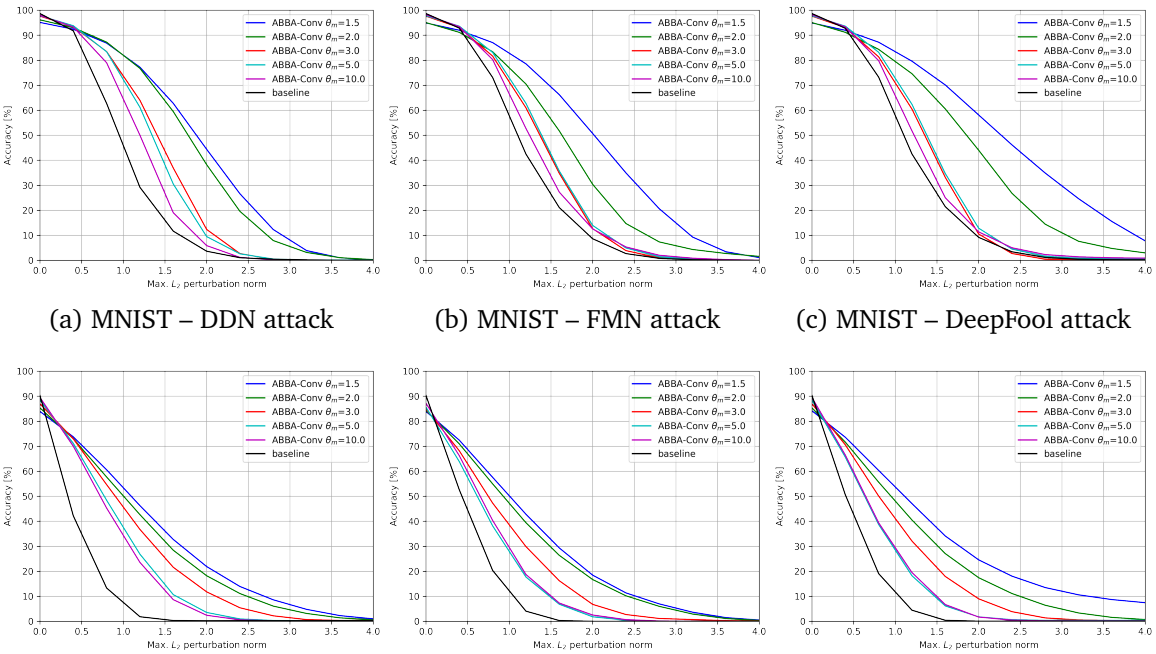
(d) FMNIST – DDN attack (e) FMNIST – FMN attack (f) FMNIST – DeepFool attack

Figure 2: *Accuracy vs. Perturbation for different Lipschitz constants – Dense Architecture*.

work, as well as other training details, are provided in Appendix SM9. For all experiments, the input images were scaled in the $[-1, 1]$ interval.

ABBA networks vs. non-negative networks. First, we compare our non-negative ABBA networks with standard ones trained under non-negativity constraints. We consider standard neural networks having the same number of parameters as their ABBA equivalent. The results are summarized in Table 1, indicating that ABBA neural networks yield far superior results, in all cases. In the case of fully connected architectures, the difference in terms of accuracy is around $\sim 3\%$ and $\sim 5\%$ for MNIST and FMNIST, respectively. The difference is even higher when we consider Conv architectures (e.g. $\sim 5\%$, $\sim 7\%$, $\sim 31\%$ and $\sim 32\%$ for MNIST, FMNIST, RPS, and CelebA, respectively). This shows that standard non-negative convolutional kernels are often suboptimal for extracting relevant information from image data. On the other hand, training standard neural networks having arbitrary-signed weights gives results very similar to their ABBA equivalents in all the cases, showing that ABBA networks do not suffer from these shortcomings. These results are in agreement with Proposition 3.6.

Stability vs. Performance. According to the "no free lunch" theorem [51], stability guarantees may impact the system performance on clean data. In this work, we train several models by following the approach described in Section 5. The Lipschitz constant of the network is varied in an effort to find the optimal trade-off between robustness and classification accuracy. This compromise is usually use-case specific, depending on the architecture complexity and on the dataset particularities, so the tightness of the imposed stability bound must be chosen accordingly. In our experiments, we limited the maximum Lipschitz constant we impose, so that the drop in performance does not exceed 5% of



(a) MNIST – DDN attack (b) MNIST – FMN attack (c) MNIST – DeepFool attack

(d) FMNIST – DDN attack (e) FMNIST – FMN attack (f) FMNIST – DeepFool attack

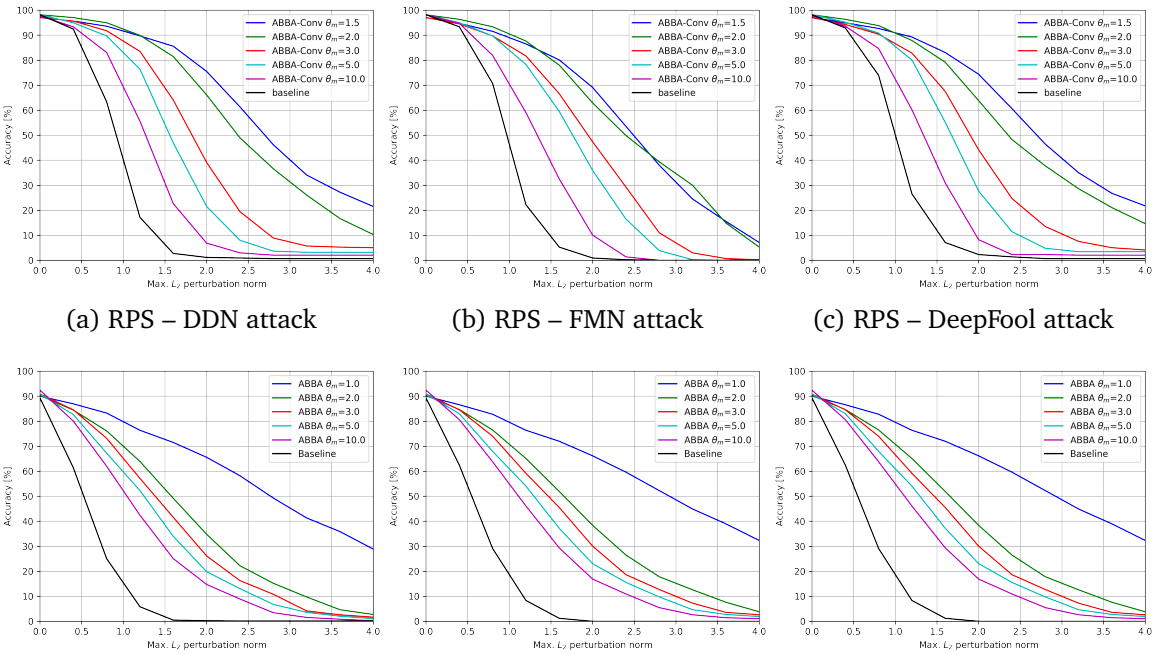
Figure 3: *Acc. vs. Perturbation for different Lipschitz constants ABBA Conv Architectures – MNIST and FMNIST*.

the baseline model accuracy (i.e., the model trained without robustness constraints).

Adversarial attack validation. We train several robust ABBA models by varying the global Lipschitz bound $\overline{\theta}_m$. We then evaluate their robustness against inputs corrupted with different levels of adversarial perturbations, by studying their influence on the overall performance of the system. To create the adversarial perturbations, we use three white-box attackers as described next. **DDN** [42] is a gradient-based ℓ_2 adversarial attack method that seeks to decouple the direction and norm of the additive perturbation. By doing so, this attack is able to generate effective examples, while requiring fewer iterations than other methods. **DeepFool** [33] considers a linear approximation to the model and refines the attack sample iteratively, by selecting the point that would cross the decision boundary with minimal effort in the logit space. The **FMN** [39] attack improves the approach in DDN by introducing adaptive norm constraints on the perturbation, in order to balance the trade-off between the magnitude of the perturbation and the level of miss-classification. This results in a powerful attack that is able to generate adversarial examples with small perturbation levels.

For a given maximum ℓ_2 perturbation norm, we ran each attack for 300 steps, using the default hyperparameters for each of the three attackers. Although all images are normalized in the $[-1, 1]$ range, we report the robust accuracy w.r.t. ℓ_2 perturbation measured in $[0, 1]$ range, which is the common practice in the literature.

The results are summarized in Figures 2, 3, and 4 which show the robustness of MNIST, FMNIST, RPS, and CelebA ABBA models, w.r.t. increasing ℓ_2 norm perturbation, generated with DDN, FMN,



(a) RPS – DDN attack (b) RPS – FMN attack (c) RPS – DeepFool attack

(d) CelebA – DDN attack (e) CelebA – FMN attack (f) CelebA – DeepFool attack

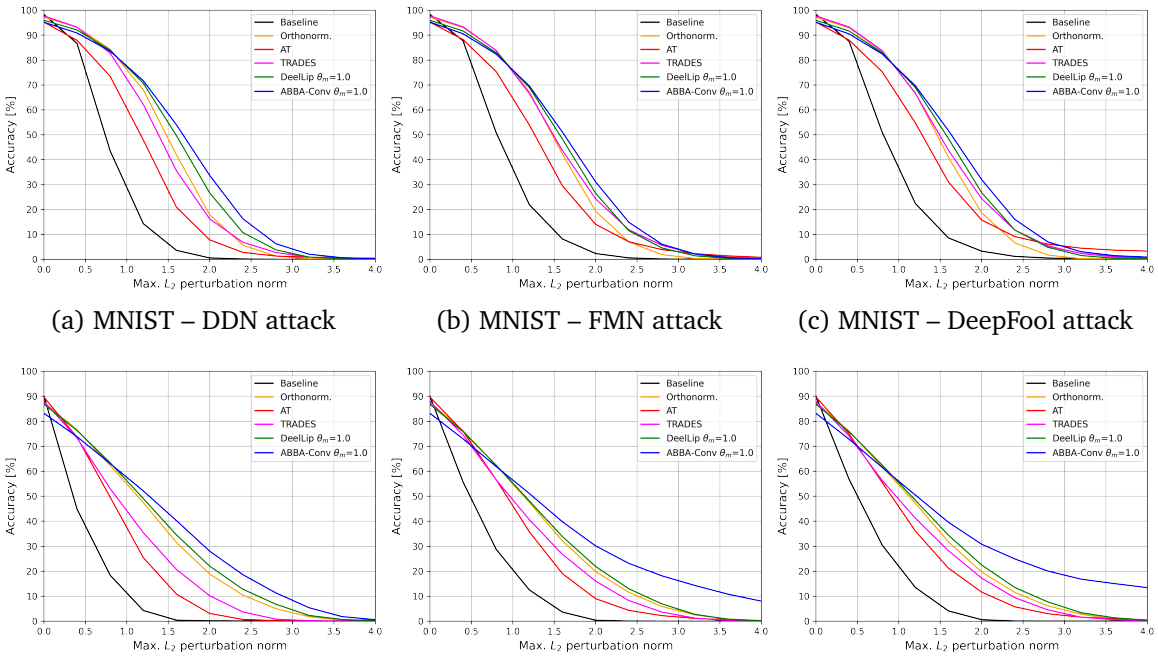
Figure 4: *Acc. vs. Perturbation for different Lipschitz constants ABBA Conv Architectures – RPS and CelebA.*

and DeepFool attacks. A baseline model, trained without stability constraints and arbitrary-signed weights, is provided as a reference. These graphs could be interpreted as the expected performance of the model if the attack is allowed to influence the input image with an ℓ_2-norm less than ϵ, where the level of perturbation ϵ varies. For a better understanding of the adversarial perturbation effect, some visual examples of the attacked inputs, for all the datasets, are presented in Appendix SM10.

It can be observed that our robust ABBA models are significantly less affected by adversarial inputs than the undefended baseline. This demonstrates that carefully controlling the Lipschitz constant during training improves the network stability against adversarial attacks. Naturally, as the imposed bound gets lower, the system becomes more robust. Although the difference in robustness between similar values of the Lipschitz constant depends on the intrinsic structure of the dataset, our results show that a good trade-off between robustness and performance can be achieved in all cases.

Comparison with other defense strategies. In the following, we compare our method for training robust models using ABBA networks with other defense strategies. *Deel-Lip*, developed in [45], is a popular Lipschitz-based approach, which uses the spectral normalization of each layer to offer robustness certificates during training. We have also made comparisons with another popular technique to ensure 1-Lipschitz weights, via *orthonormalization* [2, 8]. For our experiments, we trained models from scratch, using the implementation provided by [2] [6], in order to enforce an

[6] https://github.com/cemanil/LNets/tree/master



(a) MNIST – DDN attack (b) MNIST – FMN attack (c) MNIST – DeepFool attack

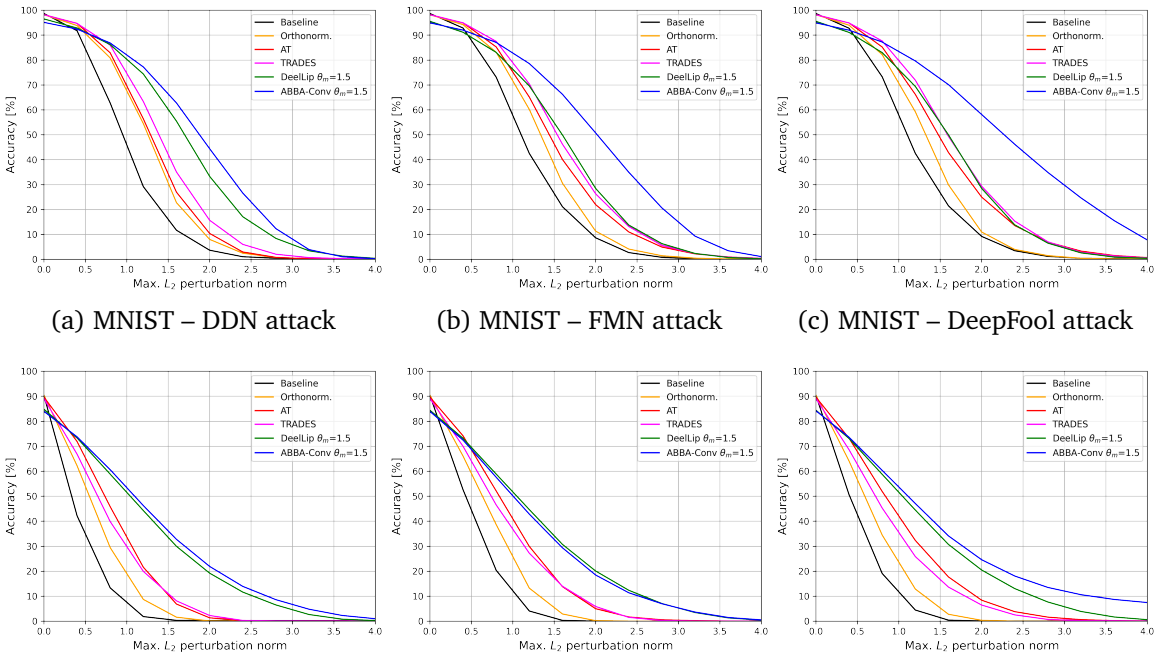
(d) FMNIST – DDN attack (e) FMNIST – FMN attack (f) FMNIST – DeepFool attack

Figure 5: *Comparisons with other defense techniques Dense Architectures*

l_2-norm equal to 1 using *Bjorck Orthonormalization*, while also preserving the gradient norm through *GroupSort*. *TRADES* [59] introduces a robustness regularization term into the training objective. This regularizer encourages the network to have similar predictions on both the original input and its adversarial counterparts. On the other hand, *Adversarial Training* (AT) implies augmenting the training data with adversarial samples, increasing the network generalization capabilities to different input alterations. However, this technique offers weak theoretical stability guarantees, as it is mainly dependent on the strength of the adversary used during training.

For all experiments regarding AT, we used Projected Gradient Descent (PGD) attack to generate the adversarial samples with a perturbation level $\epsilon = 0.5$ and then employed the scheduling strategy introduced by [32]. Concerning TRADES, we set $\lambda = 1$ for MNIST and FMNIST, and $\lambda = 1/2$ for RPS and CelebA datasets. For all the presented techniques we considered the equivalent baseline to each ABBA network.

Comparisons, in the same adversarial set-up as before, are depicted in Figures 5, 6, and 7. We observe that using our theoretically certified Lipschitz bound yields models which are generally more robust than AT and TRADES. For simple datasets, such as MNIST and FMNIST, robust ABBA and Deel-lip models exhibit similar behavior for low-magnitude adversarial attacks, but as we increase the maximum perturbation ϵ, our method performs better. In the case of real-world datasets (RPS and CelebA), ABBA models exhibit high robustness properties against all the tested attacks, showing that our approach allows us to train neural models to reach great stability properties, without losing their generalization power.



(a) MNIST – DDN attack (b) MNIST – FMN attack (c) MNIST – DeepFool attack

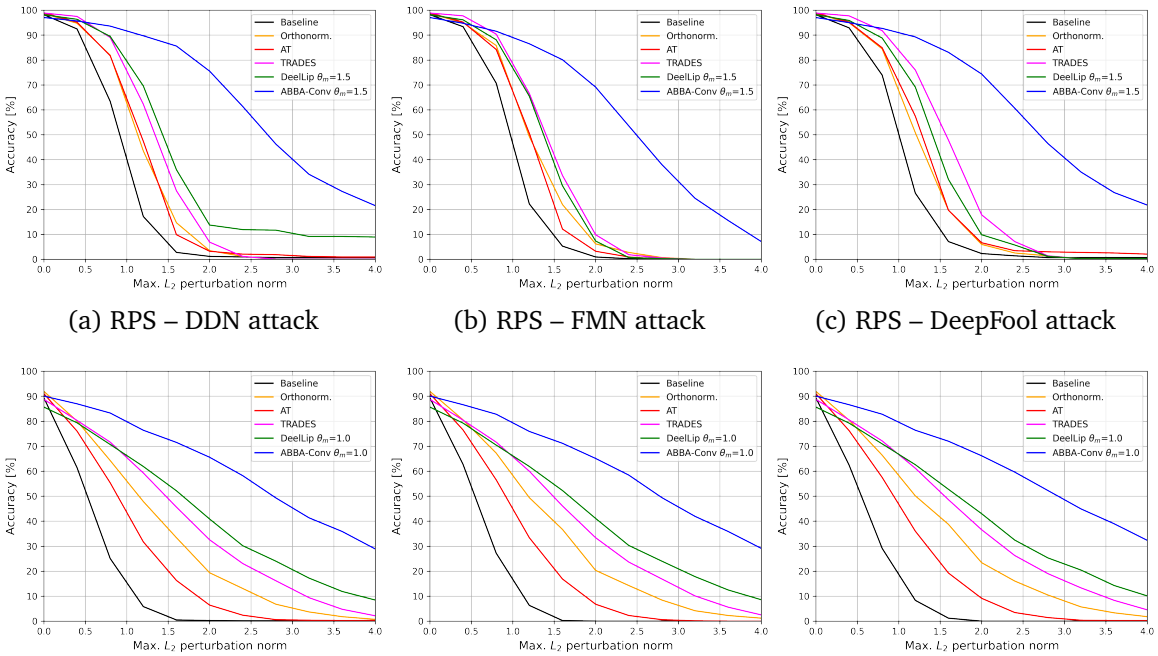
(d) FMNIST – DDN attack (e) FMNIST – FMN attack (f) FMNIST – DeepFool attack

Figure 6: *Comparisons with other defense techniques Conv Architectures – MNIST and FMNIST*.

Limitations. The main limitation of our method is that non-negative ABBA operators require more parameters to meet the universal approximation conditions. More precisely, for given depth m and number of neurons $(N_i)_{0\leqslant i\leqslant m}$ per layer, a network $\widetilde{T} \in \mathcal{N}_{m,\mathcal{A}}$ has the same number of inputs and outputs as the standard feedfoward network T in Model 3.1. All the layers of $\widetilde{T}$, except the first one, have however twice more inputs than T. Exact training times are showcased in SM11. Because of the ABBA structure of the weight matrices in (3.11), the maximum number of parameters of $\widetilde{T}$ is $2(N_0^2 + \sum_{i=1}^m N_i(N_{i-1}+1) + N_m^2)$ while the number of parameters of T is $\sum_{i=1}^m N_i(N_{i-1}+1)$. By storing $\widetilde{W}_1' = \widetilde{W}_1\widetilde{W}_0 \in \mathbb{R}^{(2N_1)\times N_0}$ instead of $\widetilde{W}_1$ and $\widetilde{W}_0$ separately, the maximum number of parameters is reduced to $2(\sum_{i=1}^m N_i(N_{i-1}+1) + N_m^2)$. Moreover, since the weights are non-negative, the model does not necessarily require signed representations storage, so the memory space occupied by $\widetilde{T}$ could also be reduced.

While our method does not provide any certification regarding the accuracy of the classifier in adversarial environments, it delivers a certified value for the Lipschitz constant of the network.

7. Conclusions. In this paper, we introduce ABBA networks, a novel class of neural networks where the majority of weights are non-negative. We demonstrate that these networks are universal approximators, possessing all the expressive properties of conventional signed neural architectures. Additionally, we unveil their remarkable algebraic characteristics, enabling us to derive precise Lipschitz bounds for both fully-connected and convolutive operators. Leveraging these bounds, we construct robust neural networks suitable for various classification tasks.



(a) RPS – DDN attack (b) RPS – FMN attack (c) RPS – DeepFool attack

(d) CelebA – DDN attack (e) CelebA – FMN attack (f) CelebA – DeepFool attack

Figure 7: *Comparisons with other defense techniques Conv Architectures – RPS and CelebA.*

The main advantage of ABBA networks is that they enjoy all the properties of non-negative networks (e.g., they are easier to interpret and less prone to overfitting), without suffering from the shortcomings of standard ones (e.g., lack of expressivity). Additionally, we showed that ABBA structures allow tight Lipschitz bounds to be estimated, without requiring to solve an NP-hard problem as for conventional neural networks.

For future research, it would be intriguing to explore the application of ABBA networks in regression problems, where controlling the Lipschitz constant may present more challenges. Also, extending our theoretical bounds to different structures, such as recurrent or attention-based networks, holds promise for further advancements. Moreover, it would be interesting to study if using Lipschitz-constrained ABBA neural networks can improve certified robustness strategies like GloRoNets [27].

Finally, we recognize the necessity of investigating the scalability of the proposed training method to deep architectures. One of the main hurdles in this endeavor is the increased number of parameters that deep ABBA architectures entail.

Acknowledgments. We would like to thank Dr. Jérôme Rony and Prof. Ismail Ben Ayed from ETS Montréal for fruitful discussions and insightful suggestions.

Appendix A. Proof of Proposition 3.6. For every $i \in \{1,\dots,m\}$, let $x_i = T_i(x_{i-1})$ where $x_0 \in \mathbb{R}^{N_0}$ is an arbitrary input of network T and $x_m \in \mathbb{R}^{N_m}$ its corresponding output. By using the symmetry properties of the activation operators, for every $i \in \{1,\dots,m\}$, we have

$$\begin{aligned}(\forall i \in \{1,\dots,m\}) \quad x_i &= R_i(W_i x_{i-1} + b_i)\\ &= R_i(W_i^+ x_{i-1} - W_i^- x_{i-1} + b_i), \qquad \text{(A.1)}\\ -x_i &= -R_i(W_i x_{i-1} + b_i)\\ &= R_i(-W_i x_{i-1} - b_i + c_i) - d_i\\ &= R_i(W_i^- x_{i-1} - W_i^+ x_{i-1} - b_i + c_i) - d_i. \qquad \text{(A.2)}\end{aligned}$$

By making use of notation (3.14), (A.1) and (A.2) can be rewritten more concisely as

$$(\forall i \in \{1,\dots,m\}) \quad \begin{bmatrix} x_i \\ -x_i \end{bmatrix} = \widetilde{R}_i\Big(\begin{bmatrix} W_i^+ & W_i^- \\ W_i^- & W_i^+ \end{bmatrix} \begin{bmatrix} x_{i-1} \\ -x_{i-1} \end{bmatrix} + \begin{bmatrix} b_i \\ c_i - b_i \end{bmatrix} \Big) - \begin{bmatrix} 0 \\ d_i \end{bmatrix}. \qquad \text{(A.3)}$$

Let us define, for every $i \in \{0,\dots,m\}$,

$$\widetilde{x}_i = \begin{bmatrix} x_i \\ -x_i + d_i \end{bmatrix}. \qquad \text{(A.4)}$$

Altogether (3.16), (3.11), (3.15), and (A.3) yield

$$\begin{aligned}(\forall i \in \{1,\dots,m\}) \quad \widetilde{x}_i &= \widetilde{R}_i \left(\begin{bmatrix} W_i^+ & W_i^- \\ W_i^- & W_i^+ \end{bmatrix} \Big(\widetilde{x}_{i-1} - \begin{bmatrix} 0 \\ d_{i-1} \end{bmatrix} \Big) + \begin{bmatrix} b_i \\ c_i - b_i \end{bmatrix} \right)\\ &= \widetilde{R}_i\Big(\widetilde{W}_i \widetilde{x}_{i-1} + \widetilde{b}_i \Big). \qquad \text{(A.5)}\end{aligned}$$

This shows that, if $(\widetilde{T}_i)_{1 \leqslant i \leqslant m}$ are given by (3.8), $\widetilde{x}_m = \widetilde{T}_m \cdots \widetilde{T}_1(\widetilde{x}_0)$. By using the forms of $\widetilde{W}_0$ and $\widetilde{W}_{m+1}$ in (3.13), we deduce that

$$x_m = \widetilde{W}_{m+1} \widetilde{x}_m + \widetilde{b}_{m+1} = \widetilde{T}(x_0). \qquad \text{(A.6)}$$

Appendix B. Proof for Proposition 3.8. According to [53] [10, Proposition 5.5],

$$\vartheta_m = \sup_{\substack{\Lambda_1 \in \mathscr{D}_{\{-1,1\}}^{2N_1} \\ \vdots \\ \Lambda_m \in \mathscr{D}_{\{-1,1\}}^{2N_m}}} \|\widetilde{W}_{m+1} \Lambda_m \widetilde{W}_m \cdots \Lambda_1 \widetilde{W}_1 \widetilde{W}_0\|. \qquad \text{(B.1)}$$

where, for every $i \in \{1,\dots,m\}$, $\mathscr{D}_{\{-1,1\}}^{2N_i}$ designates the space of diagonal matrices of size $(2N_i) \times (2N_i)$ with diagonal entries equal to -1 or 1. For every $(\Lambda_1,\dots,\Lambda_m) \in \mathscr{D}_{\{-1,1\}}^{2N_1} \times \cdots \times \mathscr{D}_{\{-1,1\}}^{2N_m}$,

$$\|\widetilde{W}_{m+1} \Lambda_m \widetilde{W}_m \cdots \Lambda_1 \widetilde{W}_1 \widetilde{W}_0\| \leqslant \|\widetilde{W}_{m+1}\| \|\widetilde{W}_m \Lambda_{m-1} \widetilde{W}_{m-1} \cdots \Lambda_2 \widetilde{W}_2 \Lambda_1 \widetilde{W}_1\| \|\widetilde{W}_0\|. \qquad \text{(B.2)}$$

On the other hand, for every $i \in \{1,\dots,m\}$, $\widetilde{W}_i \in [0,+\infty[^{(2N_i)\times(2N_{i-1})}$. It then follows from [10, Proposition 5.10] that

$$\sup_{\substack{\Lambda_1 \in \mathscr{D}_{\{-1,1\}}^{2N_1} \\ \vdots \\ \Lambda_{m-1} \in \mathscr{D}_{\{-1,1\}}^{2N_{m-1}}}} \|\widetilde{W}_m \Lambda_{m-1} \widetilde{W}_{m-1} \cdots \Lambda_2 \widetilde{W}_2 \Lambda_1 \widetilde{W}_1\| = \|\widetilde{W}_m \widetilde{W}_{m-1} \cdots \widetilde{W}_2 \widetilde{W}_1\|. \qquad \text{(B.3)}$$

According to Proposition 3.3(iii), $\widetilde{W}_m\widetilde{W}_{m-1}\cdots\widetilde{W}_2\widetilde{W}_1 \in \mathcal{A}_{N_m,N_0}$. Since this matrix has nonnegative elements, we deduce from Proposition 3.3(vii) that

$$\begin{aligned}\vartheta_m &\leqslant \|\widetilde{W}_{m+1}\|\,\|\widetilde{W}_m\cdots\widetilde{W}_1\|\,\|\widetilde{W}_0\| \\ &= \|\widetilde{W}_{m+1}\|\,\|\mathfrak{S}(\widetilde{W}_m\cdots\widetilde{W}_1)\|\,\|\widetilde{W}_0\| \\ &= \|\widetilde{W}_{m+1}\|\,\|\mathfrak{S}(\widetilde{W}_m)\cdots\mathfrak{S}(\widetilde{W}_1)\|\,\|\widetilde{W}_0\|, \end{aligned} \tag{B.4}$$

where the last equality is also a consequence of Proposition 3.3(iii). This leads to the Lipschitz bound in (3.19).

Appendix C. Proof of Theorem 4.1.

Before giving the proof of our main result, we will introduce a link between the Fourier and the spatial representations for a nonnegative convolutional kernel.

Lemma C.1. *Let $(c,c') \in (\mathbb{N}\setminus\{0\})^2$ and let*

$$(\forall \mathbf{n}\in\mathbb{Z}^d)\quad \boldsymbol{H}(\mathbf{n}) = \big(h_{q,p}(\mathbf{n})\big)_{1\leqslant q\leqslant c',1\leqslant p\leqslant c} \in [0,+\infty[^{c'\times c} \tag{C.1}$$

where, for every $p\in\{1,\ldots,c\}$ and $q\in\{1,\ldots,c'\}$, $h_{q,p}\in\ell^1(\mathbb{Z}^d)$ [7]. Then, the Fourier transform $\widehat{\boldsymbol{H}}$ of $\big(\boldsymbol{H}(\mathbf{n})\big)_{\mathbf{n}\in\mathbb{Z}^d}$ is such that

$$\sup_{\boldsymbol{\nu}\in[0,1]^d} \|\widehat{\boldsymbol{H}}(\boldsymbol{\nu})\| = \Big\| \sum_{\mathbf{n}\in\mathbb{Z}^d} \boldsymbol{H}(\mathbf{n})\Big\|. \tag{C.2}$$

Proof. For every $\boldsymbol{\nu}\in[0,1]^d$,

$$\widehat{\boldsymbol{H}}(\boldsymbol{\nu}) = \sum_{\mathbf{n}\in\mathbb{Z}^d} \boldsymbol{H}(\mathbf{n}) \exp\big(-\imath 2\pi\pi \mathbf{n}^\top\boldsymbol{\nu}\big). \tag{C.3}$$

For every $\mathbf{u}=[u_1,\ldots,u_c]^\top\in\mathbb{C}^c$, by using the triangle inequality,

$$\begin{aligned}\|\widehat{\boldsymbol{H}}(\boldsymbol{\nu})\mathbf{u}\|^2 &= \sum_{q=1}^{c'}\Big|\sum_{p=1}^{c}[\boldsymbol{H}(\boldsymbol{\nu})]_{q,p}u_p\Big|^2 \\ &= \sum_{q=1}^{c'}\Big|\sum_{p=1}^{c}\sum_{\mathbf{n}\in\mathbb{Z}^d} h_{q,p}(\mathbf{n})\exp\big(-\imath 2\pi\mathbf{n}^\top\boldsymbol{\nu}\big)u_p\Big|^2 \\ &\leqslant \sum_{q=1}^{c'}\Big(\sum_{p=1}^{c}\sum_{\mathbf{n}\in\mathbb{Z}^d} h_{q,p}(\mathbf{n})|u_p|\Big)^2 \\ &= \Big\|\sum_{\mathbf{n}\in\mathbb{Z}^d}\boldsymbol{H}(\mathbf{n})|\mathbf{u}|\Big\|^2 \\ &\leqslant \Big\|\sum_{\mathbf{n}\in\mathbb{Z}^d}\boldsymbol{H}(\mathbf{n})\Big\|^2\,\big\||\mathbf{u}|\big\|^2 \\ &= \Big\|\sum_{\mathbf{n}\in\mathbb{Z}^d}\boldsymbol{H}(\mathbf{n})\Big\|^2\,\|\mathbf{u}\|^2, \end{aligned} \tag{C.4}$$

[7] $\ell^1(\mathbb{Z}^d)$ is the space of summable d-dimensional sequences

where $|\mathbf{u}|$ denotes the vector of moduli of the components of vector $\mathbf{u}$. This shows that

$$\|\widehat{\boldsymbol{H}}(\boldsymbol{\nu})\| \leqslant \Big\| \sum_{\mathbf{n}\in\mathbb{Z}^d} \boldsymbol{H}(\mathbf{n}) \Big\|. \tag{C.5}$$

and, consequently,

$$\sup_{\boldsymbol{\nu}\in[0,1]^d} \|\widehat{\boldsymbol{H}}(\boldsymbol{\nu})\| \leqslant \Big\| \sum_{\mathbf{n}\in\mathbb{Z}^d} \boldsymbol{H}(\mathbf{n}) \Big\|. \tag{C.6}$$

In addition, the upper bound is attained since

$$\widehat{\boldsymbol{H}}(\mathbf{0}) = \sum_{\mathbf{n}\in\mathbb{Z}^d} \boldsymbol{H}(\mathbf{n}). \tag{C.7}$$

■

Next, we derive the proof for Theorem 4.1 in light of Lemma C.1.

Proof. $\boldsymbol{W}$ being the impulse response of the MIMO filter with the frequency response given by (SM4.1), it follows from Noble identities [52] that $\mathcal{W}$ is equivalent to a convolution with $\boldsymbol{W}$ followed by a decimation by a factor σ_m. Let $\mathbf{x} \in \mathcal{H}_0$. Let the σ_m-polyphase representation of $\mathbf{x}$ (resp. $\boldsymbol{W}$) be defined as

$$\big(\forall \mathbf{j} \in \mathbb{S}(\sigma_m)\big)(\forall \mathbf{n} \in \mathbb{Z}^d) \quad \mathbf{x}^{(\mathbf{j})}(\mathbf{n}) = \mathbf{x}(\sigma_m \mathbf{n} - \mathbf{j}) \tag{C.8}$$

$$\Big(\text{resp. } \boldsymbol{W}^{(\mathbf{j})}(\mathbf{n}) = \boldsymbol{W}(\sigma_m \mathbf{n} + \mathbf{j})\Big). \tag{C.9}$$

Then, as a result of multirate digital filtering, $\mathbf{y} = \mathcal{W}\mathbf{x}$ if and only if

$$\mathbf{y} = \sum_{\mathbf{j}\in\mathbb{S}(\sigma_m)} \boldsymbol{W}^{(\mathbf{j})} * \mathbf{x}^{(\mathbf{j})}. \tag{C.10}$$

This sum of MIMO convolutions can be reformulated as a single one

$$\mathbf{y} = \boldsymbol{H} * \mathbf{e}, \tag{C.11}$$

where $\boldsymbol{H}$ is the $c_m \times \sigma_m^d c_0$ MIMO impulse response obtained by stacking rowwise the polyphase MIMO impulse responses $(\boldsymbol{W}^{(\mathbf{j})})_{\mathbf{j}\in\mathbb{S}(\sigma_m)}$ and $\mathbf{e}$ is the $\sigma_m^d c_0$-component d-dimensional signal obtained by stacking columnwise the polyphase signal components $(\mathbf{x}^{(\mathbf{j})})_{\mathbf{j}\in\mathbb{S}(\sigma_m)}$. For example, if $d = 2$, we have

$$\begin{aligned}&(\forall \mathbf{n} \in \mathbb{Z}^2)\\ &\boldsymbol{H}(\mathbf{n}) = [\boldsymbol{W}^{(0,0)}(\mathbf{n}), \ldots, \boldsymbol{W}^{(\sigma_m-1,0)}(\mathbf{n}), \ldots, \boldsymbol{W}^{(0,\sigma_m-1)}(\mathbf{n}), \ldots, \boldsymbol{W}^{(\sigma_m-1,\sigma_m-1)}(\mathbf{n})]\end{aligned} \tag{C.12}$$

and

$$(\forall \mathbf{n} \in \mathbb{Z}^2) \quad \mathbf{e}(\mathbf{n}) = \begin{bmatrix} \mathbf{x}^{(0,\sigma_m-1)}(\mathbf{n}) \\ \vdots \\ \mathbf{x}^{(\sigma_m-1,\sigma_m-1)}(\mathbf{n}) \\ \vdots \\ \mathbf{x}^{(\sigma_m-1,\sigma_m-1)}(\mathbf{n}) \\ \vdots \\ \mathbf{x}^{(\sigma_m-1,\sigma_m-1)}(\mathbf{n}) \end{bmatrix}. \tag{C.13}$$

Note that, according to (C.8),

$$\begin{aligned}\|\mathbf{e}\|^2 &= \sum_{\mathbf{n}\in\mathbb{Z}^d} \|\mathbf{e}(\mathbf{n})\|^2 \\ &= \sum_{\mathbf{n}\in\mathbb{Z}^d} \sum_{\mathbf{j}\in\mathbb{S}(\sigma_m)} \|\mathbf{x}^{(\mathbf{j})}(\mathbf{n})\|^2 \\ &= \sum_{\mathbf{n}\in\mathbb{Z}^d} \|\mathbf{x}(\mathbf{n})\|^2 \\ &= \|\mathbf{x}\|^2. \qquad \text{(C.14)}\end{aligned}$$

This equality and (C.11) imply that

$$\|\mathcal{W}\| = \sup_{\boldsymbol{\nu}\in[0,1]^d} \|\widehat{\boldsymbol{H}}(\boldsymbol{\nu})\|. \qquad \text{(C.15)}$$

We thus deduce from Lemma C.1 that

$$\|\mathcal{W}\| = \Big\| \sum_{\mathbf{n}\in\mathbb{Z}^d} \boldsymbol{H}(\mathbf{n}) \Big\|. \qquad \text{(C.16)}$$

On the other hand, by using (C.9),

$$\begin{aligned}&\Big\| \sum_{\mathbf{n}\in\mathbb{Z}^d} \boldsymbol{H}(\mathbf{n}) \Big\| \\ &= \Big\| \Big(\sum_{\mathbf{n}\in\mathbb{Z}^d} \boldsymbol{H}(\mathbf{n}) \Big) \Big(\sum_{\mathbf{n}\in\mathbb{Z}^d} \boldsymbol{H}(\mathbf{n}) \Big)^\top \Big\|^{1/2} \\ &= \Big\| \sum_{\mathbf{j}\in\mathbb{S}(\sigma_m)} \Big(\sum_{\mathbf{n}\in\mathbb{Z}^d} \boldsymbol{W}^{(\mathbf{j})}(\mathbf{n}) \Big) \Big(\sum_{\mathbf{n}\in\mathbb{Z}^d} \boldsymbol{W}^{(\mathbf{j})}(\mathbf{n}) \Big)^\top \Big\|^{1/2} \\ &= \Big\| \sum_{\mathbf{j}\in\mathbb{S}(\sigma_m)} \overline{\boldsymbol{W}}^{(\mathbf{j})} \big(\overline{\boldsymbol{W}}^{(\mathbf{j})}\big)^\top \Big\|^{1/2}. \qquad \text{(C.17)}\end{aligned}$$

■

Appendix D. Proof of Theorem 4.2.

This result is a consequence of Theorem 4.1, which provides a Lipschitz bound for nonnegative convolutional neural networks. By following a similar reasoning to Appendix B, a Lipschitz constant of the ABBA network is

$$\theta_m = \|\widetilde{W}_{m+1}\| \|\widetilde{W}_m \circ \ldots \circ \widetilde{W}_1\| \|\|\widetilde{W}_0\|. \qquad \text{(D.1)}$$

Let

$$\widetilde{\boldsymbol{W}} = (\widetilde{\boldsymbol{W}}_m)_{\uparrow\sigma_{m-1}} * \cdots * (\widetilde{\boldsymbol{W}}_2)_{\uparrow\sigma_1} * \widetilde{\boldsymbol{W}}_1 \qquad \text{(D.2)}$$

and let

$$(\forall \mathbf{j} \in \mathbb{S}(\sigma_m)) \quad \Omega^{(\mathbf{j})} = \sum_{\mathbf{n}\in\mathbb{Z}^d} \mathfrak{S}\big(\widetilde{\boldsymbol{W}}(\sigma_m\mathbf{n}+\mathbf{j})\big) \in [0,+\infty[^{\zeta_m\times\zeta_0}. \qquad \text{(D.3)}$$

Since $(\widetilde{W}_i)_{1\leqslant i\leqslant m}$ are convolutional operators with nonnegative kernels, it follows from Theorem 4.1 that

$$\theta_m = \|\widetilde{W}_{m+1}\| \Big\| \sum_{\mathbf{j}\in\mathbb{S}(\sigma_m)} \overline{\boldsymbol{W}}^{(\mathbf{j})} \big(\overline{\boldsymbol{W}}^{(\mathbf{j})}\big)^\top \Big\|^{1/2} \|\widetilde{W}_0\|. \tag{D.4}$$

where, for every $\mathbf{j} \in \mathbb{S}(\sigma_m)$,

$$\overline{\boldsymbol{W}}^{(\mathbf{j})} = \sum_{\mathbf{n}\in\mathbb{Z}^d} \widetilde{\boldsymbol{W}}(\sigma_m \mathbf{n} + \mathbf{j}) \in [0, +\infty[^{(2\zeta_m)\times(2\zeta_0)}. \tag{D.5}$$

On the other hand, for every $\mathbf{n} \in \mathbb{Z}^d$, $\widetilde{\boldsymbol{W}}_i(\mathbf{n})$ is an ABBA matrix. Since $\widetilde{\boldsymbol{W}}(\mathbf{n})$ is obtained by multiplication and addition of such matrices, it follows from Proposition 3.3(ii) and (iii) that it is also an ABBA matrix. We deduce that, for every $\mathbf{j} \in \mathbb{S}(\sigma_m)$, $\overline{\boldsymbol{W}}^{(\mathbf{j})}$ is an ABBA matrix and, by using Proposition 3.3(i), $\overline{\boldsymbol{W}}^{(\mathbf{j})}\big(\overline{\boldsymbol{W}}^{(\mathbf{j})}\big)^\top$ is ABBA. By invoking now Proposition 3.3(i)-(iii) and (vii), we deduce that

$$\begin{aligned}
\Big\| \sum_{\mathbf{j}\in\mathbb{S}(\sigma_m)} \overline{\boldsymbol{W}}^{(\mathbf{j})} \big(\overline{\boldsymbol{W}}^{(\mathbf{j})}\big)^\top \Big\| &= \Big\| \mathfrak{S}\Big(\sum_{\mathbf{j}\in\mathbb{S}(\sigma_m)} \overline{\boldsymbol{W}}^{(\mathbf{j})} \big(\overline{\boldsymbol{W}}^{(\mathbf{j})}\big)^\top \Big) \Big\| \\
&= \Big\| \sum_{\mathbf{j}\in\mathbb{S}(\sigma_m)} \mathfrak{S}\Big(\overline{\boldsymbol{W}}^{(\mathbf{j})}\Big) \mathfrak{S}\Big(\overline{\boldsymbol{W}}^{(\mathbf{j})}\Big)^\top \Big\| \\
&= \Big\| \sum_{\mathbf{j}\in\mathbb{S}(\sigma_m)} \Omega^{(\mathbf{j})} \big(\Omega^{(\mathbf{j})}\big)^\top \Big\|.
\end{aligned} \tag{D.6}$$

This shows that a Lipschitz constant of the ABBA network $\widetilde{T}$ is

$$\theta_m = \|\widetilde{W}_{m+1}\| \Big\| \sum_{\mathbf{j}\in\mathbb{S}(\sigma_m)} \Omega^{(\mathbf{j})} \big(\Omega^{(\mathbf{j})}\big)^\top \Big\|^{1/2} \|\widetilde{W}_0\|, \tag{D.7}$$

Similarly to the derivation of (4.12), we deduce that $\theta_m \leqslant \overline{\theta}_m$ where $\overline{\theta}_m$ is given by (4.15).

REFERENCES

[1] B. Amos, L. Xu, and J. Z. Kolter, *Input convex neural networks*, Proc. Int. Conf. Machine Learn., vol. 70, 2017, pp. 146–155.

[2] C. Anil, J. Lucas, and R. Grosse, *Sorting out Lipschitz function approximation*, Proc. Int. Conf. Machine Learn., 2019, pp. 291–301.

[3] K. D. Apostolidis and G. A. Papakostas, *A survey on adversarial deep learning robustness in medical image analysis*, Electronics, vol. 10, 2021.

[4] B. O. Ayinde and J. M. Zurada, *Deep Learning of constrained autoencoders for enhanced understanding of data*, IEEE Trans. Neural Net. and Learn. Syst., vol. 29, 2018, pp. 3969–3979.

[5] L. Béthune, T. Boissin, M. Serrurier, F. Mamalet, C. Friedrich, and A. Gonzalez Sanz, *Pay attention to your loss : understanding misconceptions about Lipschitz neural networks*, Proc. Ann. Conf. Neur. Inform. Proc. Syst., vol. 35, 2022, pp. 20077–20091.

[6] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, *Signature verification using a "siamese" time delay neural network*, Proc. Ann. Conf. Neur. Inform. Proc. Syst., vol. 6, 1993, pp. 737–744.

[7] J. Chorowski and J. M. Zurada, *Learning understandable neural networks with nonnegative weight constraints*, IEEE Trans. Neural Net. and Learn. Syst., vol. 26, 2015, pp. 62–69.

[8] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunier, *Parseval networks: Improving robustness to adversarial examples*, Proc. Int. Conf. Mach. Learn., 2017, pp. 854–863.
[9] J. Cohen, E. Rosenfeld, and Z. Kolter, *Certified adversarial robustness via randomized smoothing*, Proc. Int. Conf. Machine Learn., 2019, pp. 1310–1320.
[10] P. L. Combettes and J.-C. Pesquet, *Lipschitz certificates for layered network structures driven by averaged activation operators*, J. Math. Data Sci., vol. 2, 2020, pp. 529–557.
[11] F. Croce, M. Andriushchenko, and M. Hein, *Provable robustness of ReLU networks via maximization of linear regions*, Proc. Int. Conf. Art. Intell. and Statistics, 2019, pp. 2057–2066.
[12] I. Diakonikolas, D. M. Kane, V. Kontonis, and N. Zarifis, *Algorithms and SQ lower bounds for PAC learning one-hidden-layer ReLU networks*, Proc. Conf. Learn. Theory, 2020, pp. 1514–1539.
[13] D. Gamarnik, E. C. Kizildağ, and I. Zadik, *Self-regularity of output weights for overparameterized two-layer neural networks*, Proc. IEEE Int. Symp. Info. Theory, 2021, pp. 819–824.
[14] M. Gleize, E. Shnarch, L. Choshen, L. Dankin, G. Moshkowich, R. Aharonov, and N. Slonim, *Are you convinced? Choosing the more convincing evidence with a Siamese Network*, Proc. Ann. Meeting of the Assoc. for Comp. Linguistics, 2019, pp. 967–976.
[15] I. Goodfellow, J. Shlens, and C. Szegedy, *Explaining and harnessing adversarial examples*, Proc. Int. Conf. Learn. Represent., 2015.
[16] S. Gowal, K. Dvijotham, R. Stanforth, T. A. Mann, and P. Kohli, *A dual approach to verify and train deep networks.*, Proc. Int. Conf. Art. Intell., 2019, pp. 6156–6160.
[17] C. Guo, B. Karrer, K. Chaudhuri, and L. van der Maaten, *Bounding training data reconstruction in private (deep) learning*, Proc. Int. Conf. Machine Learn., 2022, pp. 8056–8071.
[18] K. Gupta, F. Kaakai, B. Pesquet-Popescu, J.-C. Pesquet, and F. D. Malliaros, *Multivariate Lipschitz analysis of the stability of neural networks*, Frontiers in Signal Processing, vol. 2, 2022.
[19] A. He, C. Luo, X. Tian, and W. Zeng, *A twofold siamese network for real-time object tracking*, Proc. IEEE Conf. Comput. Vis. Pattern Recogn., 2018, pp. 4834–4843.
[20] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, *Reluplex: An efficient SMT solver for verifying deep neural networks*, Proc. Int. Conf. Comp. Aided Verif., 2017, pp. 97–117.
[21] G. Khromov and S. P. Singh, *Some fundamental aspects about Lipschitz continuity of neural network functions*, arXiv:2302.10886, 2023.
[22] P. Kidger and T. Lyons, *Universal approximation with deep narrow networks*, Proc. Conf. Learn. Theory, 2020, pp. 2306–2327.
[23] G. Koch, R. Zemel, and R. Salakhutdinov, *Siamese neural networks for one-shot image recognition*, Proc. Int. Conf. Machine Learn., vol. 2, 2015.
[24] A. Kurakin, I. Goodfellow, and S. Bengio, *Adversarial machine learning at scale*, Proc. Int. Conf. Learn. Represent., 2016.
[25] M. Lecuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana, *Certified robustness to adversarial examples with differential privacy*, Proc. IEEE Symp. Security and Privacy, 2019, pp. 656–672.
[26] D. D. Lee and H. S. Seung, *Learning the parts of objects by non-negative matrix factorization*, Nature, vol. 401, 1999, pp. 788–791.
[27] K. Leino, Z. Wang, and M. Fredrikson, *Globally-robust neural networks*, Proc. Int. Conf. Mach. Learn., 2021, pp. 6212–6222.
[28] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, *Multilayer feedforward networks with a nonpolynomial activation function can approximate any function*, Neural Networks, vol. 6, 1993, pp. 861–867.
[29] B. Li, C. Chen, W. Wang, and L. Carin, *Certified adversarial robustness with additive noise*, Proc. Ann. Conf. Neur. Inform. Proc. Syst., vol. 32, 2019.
[30] Z. Li and D. Hoiem, *Learning without forgetting*, IEEE Trans. Pattern Anal. Machine Intell., vol. 40, 2017, pp. 2935–2947.
[31] Z. Liu, P. Luo, X. Wang, and X. Tang, *Deep learning face attributes in the wild*, Proc. IEEE Int. Conf. Comput. Vis., vol. 4, 2015, pp. 3730–3738.
[32] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, *Towards deep learning models resistant to adversarial attacks*, Proc. Int. Conf. Learn. Represent., 2018.
[33] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, *Deepfool: A simple and accurate method to fool deep neural networks*, Proc. IEEE Conf. Comput. Vis. Pattern Recogn., 2016, pp. 2574–2582.
[34] L. Moroney, *Rock, paper, scissors dataset*, 2019.

[35] A. Neacșu, J.-C. Pesquet, and C. Burileanu, *EMG-based automatic gesture recognition using robust neural networks (extended version)*, Preprint submitted to ACM, 2022.
[36] S. Neumayer, A. Goujon, P. Bohra, and M. Unser, *Approximation of Lipschitz functions using deep spline neural networks*, J. Math. Data Sci., vol. 5, 2023, pp. 306–322.
[37] S. Parkinson, G. Ongie, and R. Willett, *Linear neural network layers promote learning single- and multiple-index models*, arXiv:2305.15598, 2023.
[38] P. Pauli, A. Koch, J. Berberich, P. Kohler, and F. Allgöwer, *Training robust neural networks using Lipschitz bounds*, IEEE Control Systems Letters, vol. 6, IEEE, 2021, pp. 121–126.
[39] M. Pintor, F. Roli, W. Brendel, and B. Biggio, *Fast minimum-norm adversarial attacks through adaptive norm constraints*, Proc. Ann. Conf. Neur. Inform. Proc. Syst., vol. 34, 2021, pp. 20052–20062.
[40] A. Raghunathan, J. Steinhardt, and P. Liang, *Certified defenses against adversarial examples*, Proc. Int. Conf. Learn. Represent., 2018.
[41] A. Raghunathan, J. Steinhardt, and P. S. Liang, *Semidefinite relaxations for certifying robustness to adversarial examples*, Proc. Ann. Conf. Neur. Inform. Proc. Syst., vol. 31, 2018.
[42] J. Rony, L. G. Hafemann, L. S. Oliveira, I. B. Ayed, R. Sabourin, and E. Granger, *Decoupling direction and norm for efficient gradient-based ℓ_2 adversarial attacks and defenses*, Proc. IEEE Conf. Comput. Vis. Pattern Recogn., 2019, pp. 4322–4330.
[43] S. Sabour, N. Frosst, and G. E. Hinton, *Dynamic routing between capsules*, Proc. Ann. Conf. Neur. Inform. Proc. Syst., vol. 30, 2017, pp. 3856–3866.
[44] H. Salman, J. Li, I. Razenshteyn, P. Zhang, H. Zhang, S. Bubeck, and G. Yang, *Provably robust deep learning via adversarially trained smoothed classifiers*, Proc. Ann. Conf. Neur. Inform. Proc. Syst., vol. 32, 2019, pp. 11292–11303.
[45] M. Serrurier, F. Mamalet, A. González-Sanz, T. Boissin, J.-M. Loubes, and E. Del Barrio, *Achieving robustness in classification using optimal transport with hinge regularization*, Proc. IEEE Conf. Comput. Vis. Pattern Recogn., 2021, pp. 505–514.
[46] A. Shafahi, P. Saadatpanah, C. Zhu, A. Ghiasi, C. Studer, D. Jacobs, and T. Goldstein, *Adversarially robust transfer learning*, Proc. Int. Conf. Learn. Represent., 2020.
[47] S. H. Silva and P. Najafirad, *Opportunities and challenges in deep learning adversarial robustness: A survey*, arXiv:2007.00753, 2020.
[48] A. Sinha, H. Namkoong, and J. Duchi, *Certifiable distributional robustness with principled adversarial training*, Proc. Int. Conf. Learn. Represent., 2018.
[49] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, *Intriguing properties of neural networks*, Proc. Int. Conf. Learn. Represent., 2014.
[50] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, *Deepface: Closing the gap to human-level performance in face verification*, Proc. IEEE Conf. Comput. Vis. Pattern Recogn., 2014, pp. 1701–1708.
[51] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, *There is no free lunch in adversarial robustness (but there are unexpected benefits)*, arXiv:1805.12152, 2018.
[52] P. Vaidyanathan, *Multirate Systems and Filter Banks*, Prentice Hall (NJ), 1993.
[53] A. Virmaux and K. Scaman, *Lipschitz regularity of deep neural networks: analysis and efficient estimation*, Proc. Ann. Conf. Neur. Inform. Proc. Syst., vol. 31, 2018, pp. 3839–3848.
[54] Q. Wang, M. A. Powell, A. Geisa, E. Bridgeford, and J. T. Vogelstein, *Why do networks have inhibitory/negative connections?*, arXiv:2208.03211, (2022).
[55] Y. Wang, D. Zou, J. Yi, J. Bailey, X. Ma, and Q. Gu, *Improving adversarial robustness requires revisiting misclassified examples*, Proc. Int. Conf. Learn. Represent., 2020.
[56] C. Wei and J. Z. Kolter, *Certified robustness for deep equilibrium models via interval bound propagation*, Proc. Int. Conf. Learn. Represent., 2021.
[57] R. Zhai, C. Dan, D. He, H. Zhang, B. Gong, P. Ravikumar, C.-J. Hsieh, and L. Wang, *MACER: Attack-free and scalable robust training via maximizing certified radius*, Proc. Int. Conf. Learn. Represent., 2020.
[58] B. Zhang, D. Jiang, D. He, and L. Wang, *Rethinking Lipschitz neural networks and certified robustness: A Boolean Function Perspective*, Proc. Ann. Conf. Neur. Inform. Proc. Syst., vol. 35, 2022, pp. 19398–19413.
[59] H. Zhang, Y. Yu, J. Jiao, E. Xing, L. El Ghaoui, and M. Jordan, *Theoretically principled trade-off between robustness and accuracy*, Proc. Int. Conf. Machine Learn., 2019, pp. 7472–7482.

SUPPLEMENTARY MATERIALS: ABBA Neural Networks: Coping with Positivity, Expressivity, and Robustness*

Ana Neacşu†, Jean-Christophe Pesquet‡, Vlad Vasilescu†, and Corneliu Burileanu†

SM1. Symmetric activation functions. In practice, the activation operator R_i is often separable, that is it operates componentwise:

$$(\forall x = (\xi_k)_{1\leqslant k\leqslant N_i} \in \mathbb{R}^{N_i}) \quad R_i x = (\varrho_i(\xi_k))_{1\leqslant k\leqslant N_i}\,, \tag{SM1.1}$$

where, for every $k \in \{1,\dots,N_i\}$, $\varrho_i\colon \mathbb{R}\to\mathbb{R}$. Examples of odd functions allowing us to define a symmetric separable activation operators R_i with $c_i = d_i = 0$ are

- the hyperbolic tangent activation function $\rho_i = \tanh$
- the arctangent activation function $\rho_i = (2/\pi)\arctan$
- the inverse square root linear unit function $\varrho_i\colon \mathbb{R}\to\mathbb{R}\colon \xi \mapsto \xi/\sqrt{1+\xi^2}$
- the Elliot activation function $\varrho_i\colon \mathbb{R}\to\mathbb{R}\colon \xi\mapsto \xi/(1+|\xi|)$.

Some examples of separable activation operators which are non-odd are described below. The capped ReLU function is given by

$$(\forall \xi\in\mathbb{R}) \quad \rho_i(\xi) = \begin{cases} 0 & \text{if } \xi < 0 \\ \xi & \text{if } 0\leqslant \xi < \chi \\ \chi & \text{otherwise,}\end{cases} \tag{SM1.2}$$

where $\chi \in\,]0,+\infty[$. We have then $c_i = d_i = \chi 1_{N_i}$ with $1_{N_i} = [1,\dots,1]^\top \in \mathbb{R}^{N_i}$. We can also define a leaky version of this function as

$$(\forall \xi\in\mathbb{R}) \qquad \rho_i(\xi) = \begin{cases} \alpha\xi & \text{if} \quad \xi < 0, \\ \xi & \text{if} \quad 0\leqslant \xi < \chi, \\ \alpha(\xi-\chi)+\chi & \text{otherwise,}\end{cases} \tag{SM1.3}$$

where $\alpha \in\,]0,1[$ and $\chi \in\,]0,+\infty[$ are hyper-parameters.

SM2. Proof of the properties of ABBA matrices. (i)–(iii): These properties follow from basic algebra. We will just detail the proof of the third one. Let

$$M_1 = \begin{bmatrix} A_1 & B_1 \\ B_1 & A_1 \end{bmatrix} \quad \text{and} \quad M_2 = \begin{bmatrix} A_2 & B_2 \\ B_2 & A_2 \end{bmatrix}, \tag{SM2.1}$$

*Supplementary material for SIMODS MS#M158959.
https://doi.org/10.1137/23M1589591

†Speech and Dialogue Laboratory, University Politehnica of Bucharest, Romania (ana_antonia.neacsu@upb.ro, vlad.vasilescu2111@upb.ro, corneliu.burileanu@upb.ro).

‡Centre de Vision Numérique, Inria, CentraleSupélec, Gif-sur-Yvette, France (jean-christophe.pesquet@centralesupelec.fr).

where $(A_1, B_1) \in \mathbb{R}^{N_2 \times N_1}$ and $(A_2, B_2) \in \mathbb{R}^{N_3 \times N_2}$. Then

$$M_2 M_1 = \begin{bmatrix} A_2A_1 + B_2B_1 & A_2B_1 + B_2A_1 \\ A_2B_1 + B_2A_1 & A_2A_1 + B_2B_1 \end{bmatrix} \in \mathcal{A}_{N_3,N_1}. \tag{SM2.2}$$

In addition,

$$\begin{aligned} \mathfrak{S}(M_2M_1) &= A_2A_1 + B_2B_1 + A_2B_1 + B_2A_1 \\ &= (A_2 + B_2)(A_1 + B_1) \\ &= \mathfrak{S}(M_2)\mathfrak{S}(M_1). \end{aligned} \tag{SM2.3}$$

(iv): This property is a direct consequence of (ii) and (iii).
(v): Let $M = \left[\begin{smallmatrix} A & B \\ B & A \end{smallmatrix}\right] \in \mathbb{R}^{(2N_1)\times(2N_1)}$. $\lambda \in \mathbb{C}$ is an eigenvalue of M if and only if

$$\det(M - \lambda \operatorname{Id}) = 0 \quad \Leftrightarrow \quad \det\left(\begin{bmatrix} A - \lambda \operatorname{Id} & B \\ B & A - \lambda \operatorname{Id} \end{bmatrix}\right) = 0. \tag{SM2.4}$$

We have

$$\begin{bmatrix} A - \lambda \operatorname{Id} & B \\ B & A - \lambda \operatorname{Id} \end{bmatrix} \begin{bmatrix} \operatorname{Id} & -\operatorname{Id} \\ \operatorname{Id} & \operatorname{Id} \end{bmatrix} = \begin{bmatrix} A + B - \lambda \operatorname{Id} & -A + B + \lambda \operatorname{Id} \\ A + B - \lambda \operatorname{Id} & A - B - \lambda \operatorname{Id} \end{bmatrix}. \tag{SM2.5}$$

Since $A - B - \lambda \operatorname{Id}$ and $-A + B + \lambda \operatorname{Id}$ commute, we have [SM5]

$$\det\left(\begin{bmatrix} A + B - \lambda \operatorname{Id} & -A + B + \lambda \operatorname{Id} \\ A + B - \lambda \operatorname{Id} & A - B - \lambda \operatorname{Id} \end{bmatrix}\right) = 2^N \det\left((A + B - \lambda \operatorname{Id})(A - B - \lambda \operatorname{Id})\right). \tag{SM2.6}$$

Similarly

$$\det\left(\begin{bmatrix} \operatorname{Id} & -\operatorname{Id} \\ \operatorname{Id} & \operatorname{Id} \end{bmatrix}\right) = 2^N. \tag{SM2.7}$$

We deduce from (SM2.5) that

$$\begin{aligned} &\det\left(\begin{bmatrix} A - \lambda \operatorname{Id} & B \\ B & A - \lambda \operatorname{Id} \end{bmatrix}\right) = \det\left((A + B - \lambda \operatorname{Id})(A - B - \lambda \operatorname{Id})\right) \\ \Leftrightarrow \quad &\det(M - \lambda \operatorname{Id}) = \det(A + B - \lambda \operatorname{Id}) \det(A - B - \lambda \operatorname{Id}). \end{aligned} \tag{SM2.8}$$

So λ is an eigenvalue of M if and only if $\det(A + B - \lambda \operatorname{Id}) = 0$ or $\det(A - B - \lambda \operatorname{Id}) = 0$, i.e., λ is an eigenvalue of $A + B$ or $A - B$.
(vi) Let M be defined similarly to previously with $(A, B) \in (\mathbb{R}^{N_2 \times N_1})^2$. We have

$$\|M\| = \|MM^\top\|^{1/2} = \left\|\begin{bmatrix} AA^\top + BB^\top & AB^\top + BA^\top \\ AB^\top + BA^\top & AA^\top + BB^\top \end{bmatrix}\right\|^{1/2} \tag{SM2.9}$$

According to (v), the eigenvalues of $MM^\top \in \mathcal{A}_{N_2,N_2}$ are those of $AA^\top + BB^\top + AB^\top + BA^\top = (A + B)(A + B)^\top$ and $AA^\top + BB^\top - AB^\top - BA^\top = (A - B)(A - B)^\top$. The maximum

eigenvalues of the two latter matrices are $\|A+B\|^2$ and $\|A-B\|^2$, respectively. Therefore $\|M\| = \max\{\|A+B\|, \|A-B\|\}$.
(vii): In addition, if A and B have nonnegative elements,

$$\begin{aligned}\|A-B\| &= \sup_{x\in\mathbb{R}^N\setminus\{0\}} \frac{\|Ax-Bx\|}{\|x\|} \\ &\leqslant \sup_{x\in\mathbb{R}^N\setminus\{0\}} \frac{\|A|x|+B|x|\|}{\|x\|} \\ &= \sup_{a\in[0,+\infty[^N\setminus\{0\}} \frac{\|Aa+Ba\|}{\|a\|} \\ &\leqslant \|A+B\|, \end{aligned} \tag{SM2.10}$$

where $|x|$ denotes the vector whose components are the absolute values of those of vector x. We deduce from (vi) that $\|M\| = \|A+B\| = \|\mathfrak{S}(M)\|$.
(viii): We have

$$A+B = \sum_{k=1}^{K} \lambda_k u_k v_k^\top \tag{SM2.11}$$

$$A-B = \sum_{k=1}^{K} \mu_k t_k w_k^\top. \tag{SM2.12}$$

Thus

$$A = \frac{1}{2}\sum_{k=1}^{K} (\lambda_k u_k v_k^\top + \mu_k t_k w_k^\top) \tag{SM2.13}$$

$$B = \frac{1}{2}\sum_{k=1}^{K} (\lambda_k u_k v_k^\top - \mu_k t_k w_k^\top) \tag{SM2.14}$$

and we deduce that

$$\begin{aligned}\begin{bmatrix} A & B \\ B & A \end{bmatrix} &= \sum_{k=1}^{K} \frac{1}{2}\left(\lambda_k \begin{bmatrix} u_k v_k^\top & u_k v_k^\top \\ u_k v_k^\top & u_k v_k^\top \end{bmatrix} + \mu_k \begin{bmatrix} t_k w_k^\top & -t_k w_k^\top \\ -t_k w_k^\top & t_k w_k^\top \end{bmatrix}\right) \\ &= \sum_{k=1}^{K} \frac{1}{2}\left(\lambda_k \begin{bmatrix} u_k \\ u_k \end{bmatrix} \begin{bmatrix} v_k \\ v_k \end{bmatrix}^\top + \mu_k \begin{bmatrix} t_k \\ -t_k \end{bmatrix} \begin{bmatrix} w_k \\ -w_k \end{bmatrix}^\top\right). \end{aligned} \tag{SM2.15}$$

On the other hand, for every $(k,\ell) \in \{1,\dots,K\}^2$,

$$\begin{bmatrix} u_k \\ u_k \end{bmatrix}^\top \begin{bmatrix} u_\ell \\ u_\ell \end{bmatrix} = 2u_k^\top u_\ell = \begin{cases} 2 & \text{if } k=\ell \\ 0 & \text{otherwise,} \end{cases}$$

$$\begin{bmatrix} t_k \\ -t_k \end{bmatrix}^\top \begin{bmatrix} t_\ell \\ -t_\ell \end{bmatrix} = 2t_k^\top t_\ell = \begin{cases} 2 & \text{if } k=\ell \\ 0 & \text{otherwise,} \end{cases}$$

$$\begin{bmatrix} u_k \\ u_k \end{bmatrix}^\top \begin{bmatrix} t_\ell \\ -t_\ell \end{bmatrix} = 0, \tag{SM2.16}$$

which shows that $\{\frac{1}{\sqrt{2}}\left[\begin{smallmatrix} u_k \\ u_k \end{smallmatrix}\right], \frac{1}{\sqrt{2}}\left[\begin{smallmatrix} t_k \\ -t_k \end{smallmatrix}\right]\}_{1\leqslant k\leqslant K}$ is an orthonormal family of $\mathbb{R}^{2N_2}$. For similar reasons, $\{\frac{1}{\sqrt{2}}\left[\begin{smallmatrix} v_k \\ v_k \end{smallmatrix}\right], \frac{1}{\sqrt{2}}\left[\begin{smallmatrix} w_k \\ -w_k \end{smallmatrix}\right]\}_{1\leqslant k\leqslant K}$ is an orthonormal family of $\mathbb{R}^{2N_1}$. This allows us to conclude that (SM2.15) provides a singular value decomposition of $\left[\begin{smallmatrix} A & B \\ B & A \end{smallmatrix}\right]$.

(ix): The rank of $\left[\begin{smallmatrix} A & B \\ B & A \end{smallmatrix}\right]$ is equal to the number of its nonzero singular values. From the previous result, it is thus equal to the sum of the nonzero values of $A+B$ and those of $A-B$, that is the sum of the ranks of matrices $A+B$ and $A-B$.

(x): The fact that the ABBA structure is kept by matrix mappings operating elementwise is obvious. Let us thus focus on the case of spectral functions. By using the same notation as in (viii), it follows from (SM2.15) that

(SM2.17)
$$f\left(\begin{bmatrix} A & B \\ B & A \end{bmatrix}\right) = \sum_{k=1}^{K} \frac{1}{2}\left(\varphi(\lambda_k)\begin{bmatrix} u_k v_k^\top & u_k v_k^\top \\ u_k v_k^\top & u_k v_k^\top \end{bmatrix} + \varphi(\mu_k)\begin{bmatrix} t_k w_k^\top & -t_k w_k^\top \\ -t_k w_k^\top & t_k w_k^\top \end{bmatrix}\right) = \begin{bmatrix} \tilde{A} & \tilde{B} \\ \tilde{B} & \tilde{A} \end{bmatrix},$$

where

$$\tilde{A}+\tilde{B} = \sum_{k=1}^{N} \varphi(\lambda_k) u_k v_k^\top$$

(SM2.18)
$$\tilde{A}-\tilde{B} = \sum_{k=1}^{N} \varphi(\mu_k) t_k w_k^\top.$$

(xi): By using the same notation as in (3.6), The best approximation of rank less than or equal to R to a matrix M_0 in $\mathbb{R}^{(2N_2)\times(2N_1)}$ is $f(M_0)$ where f is given by (3.6) with

(SM2.19)
$$(\forall \lambda \in \mathbb{R}_+) \quad \varphi(\lambda) = \begin{cases} \lambda & \text{if } \lambda \leqslant \tilde{\lambda}_{0,[R]} \\ 0 & \text{otherwise,} \end{cases}$$

and $\tilde{\lambda}_{0,[R]}$ is the R-th eigenvalue of M_0 when these are ordered by decreasing value: $\tilde{\lambda}_{0,1} \geqslant \ldots \geqslant \tilde{\lambda}_{0,K}$. It thus follows from (x) that if $M_0 \in \mathcal{A}_{N_2,N_1}$, then $f(M_0) \in \mathcal{A}_{N_2,N_1}$.

(xii): The projection onto the spectral ball of center 0 and and radius $\rho \in]0,+\infty[$ of a matrix $M \in \mathcal{A}_{N_2,N_1}$ is given by (3.6) where

$$(\forall \xi \in \mathbb{R}) \qquad \varphi(\xi) = \min\{\xi, \rho\}.$$

The result then follows from Property (x).

Remark SM2.1. The last result can be generalized as follows. Let $\psi\colon \mathbb{R} \to]-\infty,+\infty]$ be a lower-semicontinuous function, which is proper, even, and convex, and let

(SM2.20)
$$\begin{aligned} g\colon \mathbb{R}^{(2N_2)\times(2N_1)} &\to]-\infty,+\infty] \\ M &\mapsto \sum_{i=1}^{2K} \psi(\tilde{\lambda}_k) \end{aligned}$$

where $K = \min\{N_1, N_2\}$ and $(\tilde{\lambda}_k)_{1\leqslant k\leqslant 2K}$ are the singular values of M. The proximity operator of g at $M \in \mathbb{R}^{(2N_2)\times(2N_1)}$ is [SM1, Proposition 24.68]:

$$\begin{aligned}\operatorname{prox}_g \colon M \mapsto & \operatorname*{argmin}_{P\in\mathbb{R}^{(2N_2)\times(2N_1)}} \frac{1}{2}\|P-M\|_{\mathrm{F}}^2 + g(P) \\ & = \sum_{k=1}^{2K} \operatorname{prox}_{\psi}(\tilde{\lambda}_k)\tilde{u}_k\tilde{v}_k^\top, \end{aligned} \tag{SM2.21}$$

where $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius norm. It then follows from Property (x) that, if $M \in \mathcal{A}_{N_2,N_1}$, then $\operatorname{prox}_g(M) \in \mathcal{A}_{N_2,N_1}$.

SM3. Link between Conv layers and MIMO systems. To be rigorous, let us first define the space $\mathcal{H}_{i-1}$ (resp. $\mathcal{H}_i$) in which signals $(x_p)_{1\leqslant p\leqslant\zeta_{i-1}}$ (resp. $(y_q)_{1\leqslant q\leqslant\zeta_i}$) used in (4.1) live. Typically, $\mathcal{H}_i$ is some finite-dimensional subspace of $(\ell^2(\mathbb{Z}^d))^{\zeta_i}$ where $\ell^2(\mathbb{Z}^d)$ denotes the space of square summable discrete d-dimensional fields. For the discrete convolution $*$ to be properly defined, kernels $(w_{i,q,p})_{1\leqslant p\leqslant\zeta_{i-1},1\leqslant q\leqslant\zeta_i}$ are then assumed to be summable. In practice, this assumption is satisfied since these kernels are chosen with finite size.

For $x = (x(\mathbf{n}))_{\mathbf{n}\in\mathbb{Z}^d} \in \ell^2(\mathbb{Z}^d)$, the decimation operation $(\cdot)\downarrow_{s_i}$ returns the output signal

$$(\forall \mathbf{n} \in \mathbb{Z}^d) \quad y(\mathbf{n}) = u(s_i\mathbf{n}). \tag{SM3.1}$$

Eq. (4.1) defines a MIMO (multi-input multi-output) filter that can be rexpressed in a matrix form as

$$\begin{aligned}(\forall \mathbf{n} \in \mathbb{Z}^d) \quad \mathbf{u}(\mathbf{n}) &= \sum_{\mathbf{n}\in\mathbb{Z}^d} \boldsymbol{W}_i(\mathbf{n}')\,\mathbf{x}(\mathbf{n}-\mathbf{n}') \\ &= (\boldsymbol{W}_i * \mathbf{x})(\mathbf{n}), \end{aligned} \tag{SM3.2}$$

where

$$\mathbf{u}(\mathbf{n}) = \begin{bmatrix} u_1(\mathbf{n}) \\ \vdots \\ u_{\zeta_i}(\mathbf{n}) \end{bmatrix} \in \mathbb{R}^{\zeta_j}, \quad \mathbf{x}(\mathbf{n}) = \begin{bmatrix} x_1(\mathbf{n}) \\ \vdots \\ x_{\zeta_{i-1}}(\mathbf{n}) \end{bmatrix} \in \mathbb{R}^{\zeta_{i-1}}, \tag{SM3.3}$$

and $\boldsymbol{W}_i(\mathbf{n})$ is given by (4.4). $(\boldsymbol{W}_i(\mathbf{n}))_{\mathbf{n}\in\mathbb{Z}^d}$ defines the so-called MIMO impulse response of $\mathcal{W}_i$. The MIMO impulse response of an ABBA layer is similarly given by (4.5).

These relations can also be written more concisely in the d-dimensional frequency domain[1] as

$$(\forall \boldsymbol{\nu} \in [0,1]^d) \quad \widehat{\mathbf{u}}(\boldsymbol{\nu}) = \widehat{\boldsymbol{W}}_i(\boldsymbol{\nu})\,\widehat{\mathbf{x}}(\boldsymbol{\nu}), \tag{SM3.4}$$

where

$$\widehat{\mathbf{x}}(\boldsymbol{\nu}) = \sum_{\mathbf{n}\in\mathbb{Z}^d} \mathbf{x}(\mathbf{n}) \exp(-\imath 2\pi \mathbf{n}^\top \boldsymbol{\nu}) \in \mathbb{C}^{\zeta_{j-1}}, \tag{SM3.5}$$

[1] Alternatively, we could use the d-dimensional z-transform since we are dealing with discrete-space signals.

$$\widehat{\boldsymbol{W}}_i(\boldsymbol{\nu}) = \sum_{\mathbf{n}\in\mathbb{Z}^d} \boldsymbol{W}_i(\mathbf{n}) \exp(-\imath 2\pi \mathbf{n}^\top \boldsymbol{\nu}) \in \mathbb{C}^{\zeta_{j-1}\times\zeta_j}, \tag{SM3.6}$$

and $\widehat{\boldsymbol{W}}_i$ is the frequency response of the associated MIMO filter.

Note that $\int_{[0,1]^d} \|\widehat{\mathbf{x}}(\boldsymbol{\nu})\|^2 d\boldsymbol{\nu} < +\infty$, whereas $\widehat{\boldsymbol{W}}_i$ is a continuous (hence bounded) function on $[0,1]^d$. Another useful result from sampling theory [SM6] is that the Fourier transform of $\mathbf{y} = (y_q)_{1\leqslant q\leqslant \zeta_j}$ in (4.1) is deduced from the Fourier transform of $\mathbf{u}$ by the relation

$$(\forall \boldsymbol{\nu} \in [0,1]^d) \quad \widehat{\mathbf{y}}(\boldsymbol{\nu}) = \frac{1}{s_i^d} \sum_{\mathbf{j}\in\mathbb{S}(s_i)} \widehat{\mathbf{u}}\left(\frac{\boldsymbol{\nu}+\mathbf{j}}{s_i}\right). \tag{SM3.7}$$

where

$$(\forall \sigma \in \mathbb{N}\setminus\{0\}) \quad \mathbb{S}(\sigma) = \{0,\dots,\sigma-1\}^d. \tag{SM3.8}$$

It is also worth noting that the interpolation by a factor s of $\mathbf{y}$

$$\mathbf{v} = \mathbf{y}_{\uparrow_s} \quad \Leftrightarrow \quad (\forall \mathbf{n}\in\mathbb{Z}^d) \quad \mathbf{v}(\mathbf{n}) = \begin{cases} \mathbf{y}\left(\dfrac{\mathbf{n}}{s}\right) & \text{if } \mathbf{n} \in s\,\mathbb{Z}^d \\ 0 & \text{otherwise,} \end{cases} \tag{SM3.9}$$

translates into

$$(\forall \nu \in [0,1]) \quad \widehat{\mathbf{y}}_{\uparrow_s}(\boldsymbol{\nu}) = \widehat{\mathbf{y}}(s\boldsymbol{\nu}), \tag{SM3.10}$$

in the frequency domain.

SM4. Frequency expressions of Lipschitz bounds. In this appendix, we establish frequency-based bounds of the Lipschitz constant of an m-layer convolutional neural network T.

Based on the MIMO concepts introduced in section SM3, we will introduce the following global frequency response of the network:

$$(\forall \boldsymbol{\nu} \in [0,1]^d) \quad \widehat{\boldsymbol{W}}(\boldsymbol{\nu}) = \widehat{\boldsymbol{W}}_m(\sigma_{m-1}\boldsymbol{\nu})\cdots\widehat{\boldsymbol{W}}_2(\sigma_1\boldsymbol{\nu})\widehat{\boldsymbol{W}}_1(\boldsymbol{\nu}) \in \mathbb{C}^{\zeta_m\times\zeta_0}, \tag{SM4.1}$$

where $\widehat{\boldsymbol{W}}_i$ is the frequency response associated to filter $\boldsymbol{W}_i$ (see (SM3.6)).

We have then the following result providing a frequency formula for evaluating the Lipschitz constant of a convolutional network.

Proposition SM4.1. *The quantity*

$$\theta_m = \frac{1}{\sigma_m^{d/2}} \sup_{\boldsymbol{\nu}\in[0,1/\sigma_m]^d} \left\| \sum_{\mathbf{j}\in\mathbb{S}(\sigma_m)} \widehat{\boldsymbol{W}}\left(\boldsymbol{\nu}+\frac{\mathbf{j}}{\sigma_m}\right) \widehat{\boldsymbol{W}}\left(\boldsymbol{\nu}+\frac{\mathbf{j}}{\sigma_m}\right)^{\mathsf{H}} \right\|^{1/2}. \tag{SM4.2}$$

provides a lower bound on the Lipschitz constant estimate of network T[2]*. In addition, if for every $i \in \{1,\dots,m\}$, $p\in\{1,\dots,\zeta_{i-1}\}$, and $q\in\{1,\dots,\zeta_i\}$, $w_{i,q,p} = (w_{i,q,p}(\mathbf{n}))_{\mathbf{n}\in\mathbb{Z}^d}$ is a nonnegative kernel i.e.,*

$$(\forall \mathbf{n}\in\mathbb{Z}^d) \quad w_{i,p,q}(\mathbf{n}) \geqslant 0, \tag{SM4.3}$$

then θ_m is a Lipschitz constant of T.

[2] $(\cdot)^{\mathsf{H}}$ denotes the Hermitian transpose operation.

Proof. In the considered case all activation operators are nonexpansive and they are assumed separable, except maybe at the last layer. Thus T is a special case of the networks investigated in [SM3, Section 5] for which a tight estimate of the Lipschitz constant was provided. It then follows from [SM3, Theorem 5.2] that a lower bound on this Lipschitz constant estimate is

$$\theta_m = \|\mathcal{W}_m \circ \cdots \circ \mathcal{W}_1\|. \tag{SM4.4}$$

In addition, under the additional assumption that all the kernels are nonnegative, T is an instance of the positively weighted networks investigated in [SM3, Section 5.3] and it follows from [SM3, Proposition 5.10] that θ_m is then a Lipschitz constant of T.
So the problem is to calculate the norm of the linear operator $\mathcal{W} = \mathcal{W}_m \circ \cdots \circ \mathcal{W}_1$. Each operator $\mathcal{W}_i$ with $i \in \{1, \dots, m\}$ is the composition of a d-dimensional MIMO filter with a decimator. It follows from Noble identities [SM6] that $\mathcal{W}$ reduces to cascading a $\zeta_m \times \zeta_0$ MIMO filter with frequency response $\widetilde{\boldsymbol{W}}$ with a decimation of each output by a factor σ_m. More precisely, if $\mathbf{x} \in \mathcal{H}_0$ is the input of this linear system and $\mathbf{y}$ its output, we have in the frequency domain:

$$\begin{aligned}(\forall \boldsymbol{\nu} \in [0,1]^d) \quad \widehat{\mathbf{y}}(\boldsymbol{\nu}) &= \frac{1}{\sigma_m^d} \sum_{\mathbf{j} \in \mathbb{S}(\sigma_m)} \widehat{\boldsymbol{W}}\left(\frac{\boldsymbol{\nu}+\mathbf{j}}{\sigma_m}\right) \widehat{\mathbf{x}}\left(\frac{\boldsymbol{\nu}+\mathbf{j}}{\sigma_m}\right) \\ &= \frac{1}{\sigma_m^d} \widetilde{\boldsymbol{W}}\left(\frac{\boldsymbol{\nu}}{\sigma_m}\right) \widetilde{\mathbf{x}}\left(\frac{\boldsymbol{\nu}}{\sigma_m}\right), \end{aligned} \tag{SM4.5}$$

where $\widetilde{\mathbf{x}}(\frac{\boldsymbol{\nu}}{\sigma_m})$ is a vector of dimension $\mathbb{C}^{\sigma_m^d \zeta_0}$ where the vectors $(\widehat{\mathbf{x}}((\boldsymbol{\nu}+\mathbf{j})/\sigma_m))_{\mathbf{j} \in \mathbb{S}(\sigma_m)}$ are stacked columnwise and $\widetilde{\boldsymbol{W}}(\frac{\boldsymbol{\nu}}{\sigma_m})$ is a $c_m \times \sigma_m^d \zeta_0$ matrix where the matrices $(\widehat{\boldsymbol{W}}((\boldsymbol{\nu}+\mathbf{j})/\sigma_m))_{\mathbf{j} \in \mathbb{S}(\sigma_m)}$ are stacked rowwise. For example, when $d = 2$, we have, for every $\boldsymbol{\nu} = (\nu_1, \nu_2) \in [0,1]^2$,

$$\widetilde{\mathbf{x}}(\boldsymbol{\nu}) = \begin{bmatrix} \check{\mathbf{x}}(\nu_1, \nu_2) \\ \check{\mathbf{x}}\left(\nu_1, \nu_2 + \frac{1}{\sigma_m}\right) \\ \vdots \\ \check{\mathbf{x}}\left(\nu_1, \nu_2 + \frac{\sigma_m - 1}{\sigma_m}\right) \end{bmatrix} \in \mathbb{C}^{\sigma_m^2 \zeta_0} \tag{SM4.6}$$

$$\check{\mathbf{x}}(\boldsymbol{\nu}) = \begin{bmatrix} \widehat{\mathbf{x}}(\nu_1, \nu_2) \\ \widehat{\mathbf{x}}\left(\nu_1 + \frac{1}{\sigma_m}, \nu_2\right) \\ \vdots \\ \widehat{\mathbf{x}}\left(\nu_1 + \frac{\sigma_m - 1}{\sigma_m}, \nu_2\right) \end{bmatrix} \in \mathbb{C}^{\sigma_m \zeta_0} \tag{SM4.7}$$

$$\widetilde{\boldsymbol{W}}(\boldsymbol{\nu}) = \begin{bmatrix} \check{\boldsymbol{W}}(\nu_1, \nu_2) & \check{\boldsymbol{W}}\left(\nu_1, \nu_2 + \frac{1}{\sigma_m}\right) & \dots & \check{\boldsymbol{W}}\left(\nu_1, \nu_2 + \frac{\sigma_m - 1}{\sigma_m}\right) \end{bmatrix} \in \mathbb{C}^{\zeta_m \times \sigma_m^2 \zeta_0} \tag{SM4.8}$$

$$\check{\boldsymbol{W}}(\boldsymbol{\nu}) = \begin{bmatrix} \widehat{\boldsymbol{W}}(\nu_1, \nu_2) & \widehat{\boldsymbol{W}}\left(\nu_1 + \frac{1}{\sigma_m}, \nu_2\right) & \dots & \widehat{\boldsymbol{W}}\left(\nu_1 + \frac{\sigma_m - 1}{\sigma_m}, \nu_2\right) \end{bmatrix} \in \mathbb{C}^{\zeta_m \times \sigma_m \zeta_0}. \tag{SM4.9}$$

By using now Parseval's formula,

$$\begin{aligned}
\|\mathbf{y}\|^2 &= \int_{[0,1]^d} \|\widehat{\mathbf{y}}(\boldsymbol{\nu})\|^2 d\boldsymbol{\nu} \\
&= \frac{1}{\sigma_m^{2d}} \int_{[0,1]^d} \left\| \widetilde{\boldsymbol{W}}\left(\frac{\boldsymbol{\nu}}{\sigma_m}\right) \widetilde{\mathbf{x}}\left(\frac{\boldsymbol{\nu}}{\sigma_m}\right) \right\|^2 d\boldsymbol{\nu} \\
&\leqslant \frac{1}{\sigma_m^{d}} \int_{[0,1/\sigma_m]^d} \left\| \widetilde{\boldsymbol{W}}(\boldsymbol{\nu}) \right\|^2 \|\widetilde{\mathbf{x}}(\boldsymbol{\nu})\|^2 \, d\nu_1 d\nu_2 \\
&\leqslant \frac{1}{\sigma_m^{d}} \sup_{\boldsymbol{\nu}\in[0,1/\sigma_m]^d} \left\| \widetilde{\boldsymbol{W}}(\boldsymbol{\nu}) \right\|^2 \int_{[0,1/\sigma_m]^d} \|\widetilde{\mathbf{x}}(\boldsymbol{\nu})\|^2 \, d\boldsymbol{\nu} \\
&= \frac{1}{\sigma_m^{2}} \sup_{\boldsymbol{\nu}\in[0,1/\sigma_m]^d} \left\| \widetilde{\boldsymbol{W}}(\boldsymbol{\nu}) \right\|^2 \sum_{\mathbf{j}\in\mathbb{S}(\sigma_m)} \int_{[0,1/\sigma_m]^d} \left\| \widehat{\mathbf{x}}\left(\boldsymbol{\nu} + \frac{\mathbf{j}}{\sigma_m}\right) \right\|^2 d\boldsymbol{\nu} \\
&= \frac{1}{\sigma_m^{d}} \sup_{\boldsymbol{\nu}\in[0,1/\sigma_m]^d} \left\| \widetilde{\boldsymbol{W}}(\boldsymbol{\nu}) \right\|^2 \int_{[0,1]^d} \|\widehat{\mathbf{x}}(\boldsymbol{\nu})\|^2 \, d\boldsymbol{\nu} \\
&= \frac{1}{\sigma_m^{d}} \sup_{\boldsymbol{\nu}\in[0,1/\sigma_m]^d} \left\| \widetilde{\boldsymbol{W}}(\boldsymbol{\nu}) \right\|^2 \|\mathbf{x}\|^2. \qquad \text{(SM4.10)}
\end{aligned}$$

This shows that

$$\theta_m^2 \leqslant \frac{1}{\sigma_m^{d}} \sup_{\boldsymbol{\nu}\in[0,1/\sigma_m]^d} \left\| \widetilde{\boldsymbol{W}}(\boldsymbol{\nu}) \right\|^2. \qquad \text{(SM4.11)}$$

On the other hand since $\widehat{\boldsymbol{W}}$ is continuous, $\widetilde{\boldsymbol{W}}$ is also continuous, and there exists $\widehat{\boldsymbol{\nu}} \in [0,1/\sigma_m]^d$ such that

$$\sup_{\boldsymbol{\nu}\in[0,1/\sigma_m]^d} \left\| \widetilde{\boldsymbol{W}}(\boldsymbol{\nu}) \right\| = \left\| \widetilde{\boldsymbol{W}}(\widehat{\boldsymbol{\nu}}) \right\|. \qquad \text{(SM4.12)}$$

Let us now choose, for every $\boldsymbol{\nu} \in [0,1/\sigma_m]^d$, $\widetilde{\mathbf{x}}\,(\boldsymbol{\nu}) = \alpha_\epsilon(\boldsymbol{\nu})\mathbf{u}(\boldsymbol{\nu})$ where $\mathbf{u}(\boldsymbol{\nu})$ is a unit norm eigenvector associated with the maximum eigenvalue of $\widetilde{\boldsymbol{W}}(\boldsymbol{\nu})^{\mathsf{H}}\widetilde{\boldsymbol{W}}(\boldsymbol{\nu})$, $\epsilon \in]0,+\infty[$, and

$$\alpha_\epsilon(\boldsymbol{\nu}) = \begin{cases} \frac{1}{\epsilon^{d/2}} & \text{if } (\exists \mathbf{j} \in \{-1,0,1\}^d) \ \|\boldsymbol{\nu} + \frac{\mathbf{j}}{\sigma_m} - \widehat{\boldsymbol{\nu}}\|_\infty \leqslant \frac{\epsilon}{2} \\ 0 & \text{otherwise.} \end{cases} \qquad \text{(SM4.13)}$$

Then we see that when $\epsilon \to 0$, the upper bound in (SM4.10) is reached. We conclude that

$$\theta_m = \frac{1}{\sigma_m^{d/2}} \sup_{\boldsymbol{\nu}\in[0,1/\sigma_m]^d} \left\| \widetilde{\boldsymbol{W}}(\boldsymbol{\nu}) \right\|. \qquad \text{(SM4.14)}$$

In addition, by using the relation between $\widetilde{\boldsymbol{W}}$ and $\widehat{\boldsymbol{W}}$ (i.e., (SM4.8) and (SM4.9) in the 2D case),

$$\begin{aligned}
\left\| \widetilde{\boldsymbol{W}}(\boldsymbol{\nu}) \right\|^2 &= \left\| \widetilde{\boldsymbol{W}}(\boldsymbol{\nu}) \widetilde{\boldsymbol{W}}(\boldsymbol{\nu})^{\mathsf{H}} \right\| \\
&= \left\| \sum_{\mathbf{j}\in\mathbb{S}(\sigma_m)} \widehat{\boldsymbol{W}}\left(\boldsymbol{\nu} + \frac{\mathbf{j}}{\sigma_m}\right) \widehat{\boldsymbol{W}}\left(\boldsymbol{\nu} + \frac{\mathbf{j}}{\sigma_m}\right)^{\mathsf{H}} \right\|. \qquad \text{(SM4.15)}
\end{aligned}$$

Gathering the last two equalities yields (SM4.2). ∎

When there is no decimation, i.e. the strides $(s_i)_{1\leqslant i\leqslant m}$ are all equal to 1, (SM4.2) reduces to

$$\theta_m = \sup_{\boldsymbol{\nu}\in[0,1]^d} \|\widehat{\boldsymbol{W}}_m(\boldsymbol{\nu})\cdots\widehat{\boldsymbol{W}}_2(\boldsymbol{\nu})\widehat{\boldsymbol{W}}_1(\boldsymbol{\nu})\|. \tag{SM4.16}$$

We recall that the following upper bound holds [49]:

$$\theta_m \leqslant \overline{\theta}_m = \prod_{i=1}^{m} \|\mathcal{W}_i\|. \tag{SM4.17}$$

Applying our result to the one-layer case shows that, for every $i \in \{1,\dots,m\}$,

$$\|\mathcal{W}_i\| = \frac{1}{s_i^{d/2}} \sup_{\boldsymbol{\nu}\in[0,1/s_i]^2} \left\| \sum_{\mathbf{j}\in\mathbb{S}(s_i)} \widehat{\boldsymbol{W}}_i\left(\boldsymbol{\nu}+\frac{\mathbf{j}}{s_i}\right) \widehat{\boldsymbol{W}}_i\left(\boldsymbol{\nu}+\frac{\mathbf{j}}{s_i}\right)^{\mathsf{H}} \right\|^{1/2}. \tag{SM4.18}$$

Note that the resulting upper bound in (SM4.17) gives a loose estimate of the Lipschitz constant, which has however the merit to be valid for convolutional networks having kernels with an arbitrary sign.

SM5. Numerical evaluation of the Lipschitz constant of nonnegative convolutional networks. We compare the tight bound θ_m in Theorem 4.1 with the separable one $\overline{\theta}_m$ given by (4.12) for a classic convolutional network using non-negative kernels. The results provided in Table SM1 correspond to the convolutive part of LeNet-5 [SM4]. In our experiments, we initialized the networks with randomly sampled weights drawn from a uniform distribution on $[0,1]$. Table SM1 shows the relative difference

$$\epsilon_{\mathrm{r}} = \frac{\overline{\theta}_m - \theta_m}{\overline{\theta}_m},$$

for 10 distinct noise realizations. We thus observe that the difference between the two bounds is small. Similar observations can be made on various convolutive architectures. In contrast, for fully connected networks, a separable bound is usually overpessimistic.

SM6. Lipschitz constant of average pooling. We consider the case when the i-th layer is an *average pooling* where the average is computed on patches of length L_i in each dimension and with stride s_i. For simplicity, we suppose that L_i is a multiple of s_i. The number of input and output channels is then equal, i.e. $\zeta_i = \zeta_{i-1}$. The average is calculated on each channel

Table SM1

Lipschitz bounds obtained for 10 *independent realizations of random positive initialization for LeNet-*5.

	LeNet-5									
	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
θ_m	30302.73	27734.91	30298.73	29374.35	30180.16	28632.60	30615.02	30395.67	34828.90	30097.62
$\overline{\theta}_m$	30696.07	28114.29	30860.56	29821.62	30670.05	29298.64	31152.06	30866.87	35220.36	30367.71
ϵ_{r} [%]	1.28	1.35	1.82	1.50	1.60	2.27	1.72	1.53	1.11	0.89

independently, this operation is a special case of a nonnegative convolutional layer where, for every $\mathbf{n} \in \mathbb{Z}^d$, $\boldsymbol{W}_i(\mathbf{n})$ is a diagonal matrix. The diagonal elements of this matrix are

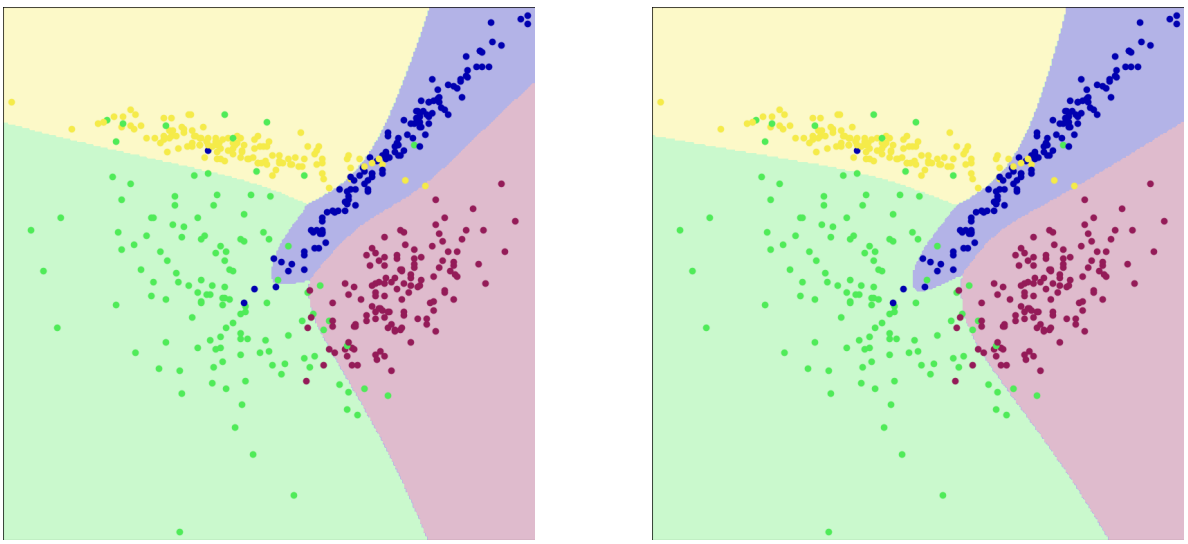
$$(\forall p \in \{1, \ldots, \zeta_i\})(\forall \mathbf{n} \in \mathbb{Z}^d) \quad w_{i,p,p}(\mathbf{n}) = \begin{cases} \frac{1}{L_i^d} & \text{if } \mathbf{n} \in [0, L_i - 1]^d \\ 0 & \text{otherwise.} \end{cases} \tag{SM6.1}$$

We deduce that, for every $\mathbf{j} \in \mathbb{S}(s_i)$, the matrix $\overline{\boldsymbol{W}}_i^{(\mathbf{j})}$ is also a diagonal matrix. More precisely, the sum in (4.13) can be restricted to values of $\mathbf{n} \in \{0, \ldots, L_i/s_i - 1\}^d$ and $\overline{\boldsymbol{W}}_i^{(\mathbf{j})} = \frac{1}{s_i^d}\mathbf{Id}$. We deduce that the Lipschitz constant of the average pooling layer is

$$\|\mathcal{W}_i\| = \Big\| \sum_{\mathbf{j} \in \mathbb{S}(s_i)} \overline{\boldsymbol{W}}_i^{(\mathbf{j})} \left(\overline{\boldsymbol{W}}_i^{(\mathbf{j})}\right)^\top \Big\|^{1/2} = \frac{1}{s_i^{d/2}}. \tag{SM6.2}$$

We see that this constant is independent of the patch size and is a decreasing function of the stride.

SM7. Expressivity of ABBA networks – simulations. For this experiment, we randomly sampled points from four distinct 2D Gaussian distributions, with different means and covariance matrices, totaling 125 2-dimensional points per class. Figure SM1 shows a comparison between decision boundaries resulting from training two models: a standard one trained conventionally and its non-negative ABBA equivalent. The two models reach a similar solution, showing that the theoretical properties proved in this paper are also observed in practical simulations.



(a) Standard model (b) ABBA model

Figure SM1. *Decision space comparison between fitting an ABBA network and a standard arbitrary-signed one.*

SM8. Constrained training of signed convolutional layers. The first and the last layers of an ABBA convolutional network have signed kernels. The norm of these layers is computed by using (SM4.18) and constrained to be less than $\overline{\theta}_{m,i,t}$ with $i \in \{0, m+1\}$. Note that (SM4.18) makes use of the frequency response $\widehat{\boldsymbol{W}}_i$ of filter $\mathcal{W}_i$. A discrete Fourier transform (DFT) is actually implemented (using 128×128 discrete frequencies). In the discrete frequency domain, the upper bound constraint is thus decomposed into 128^2 matrix norm bounds obtained by summing over s_i^2 frequencies. The projection onto each of these elementary constraint sets is computed by truncating a singular value decomposition. An additional constraint, however, is to be addressed, which is related to the fact that the kernels are of finite size. This implicitly defines a linear constraint. Projecting onto the associated vector space is simply obtained by truncating the kernel (after inverse DFT) to the desired size. The set $\mathcal{S}_{i,t}$ is thus defined as the intersection of the former matrix norm constraint set and the latter vector space. Projecting onto this intersection can be achieved by an iterative convex optimization approach. In our case, we use a Douglas-Rachford algorithm [SM2].

SM9. ABBA architectures. Table SM3 details the ABBA Dense and ABBA Conv architectures used for MNIST and FMNIST datasets, while Table SM2 shows our choices for RPS and CelebA datasets. As the ABBA layers have a specific form, their output size will be twice the number of filters. The used activation operator is the Capped Leaky ReLu (CLR) function defined in (SM1.3) for all Dense layers. For convolutional operators we employed a 3×3 kernel, using the same activation.

We used the official train-test split provided by the Tensorflow framework for both MNIST and FMNIST datasets and did not employ any augmentation strategy during training. For RPS and CelebA models, we resized the input images to 150×150, resp. 128×128, before feeding them to the network. In the case of CelebA dataset, we opted for a binary classification task on the *bald* feature. We extracted all the images containing the *bald* attribute, and we

Table SM2
ABBA Conv architectures details for RPS and CelebA datasets.

Layer type	RPS	stride	CelebA	stride
Input	$150 \times 150 \times 3$		$128 \times 128 \times 3$	
Conv2D	$150 \times 150 \times 8$	1	$128 \times 128 \times 8$	1
ABBA Conv2D + CLR	–	–	$64 \times 64 \times 8(\times 2)$	2
ABBA Conv2D + CLR	$75 \times 75 \times 32(\times 2)$	2	$32 \times 32 \times 16(\times 2)$	2
ABBA Conv2D + CLR	$37 \times 37 \times 64(\times 2)$	2	$16 \times 16 \times 32(\times 2)$	2
ABBA Conv2D + CLR	$18 \times 18 \times 128(\times 2)$	2	$8 \times 8 \times 64(\times 2)$	2
Conv2D	$18 \times 18 \times 128$	1	$8 \times 8 \times 64$	1
Global Max-Pooling2D	$128(\times 2)$		$64(\times 2)$	
ABBA Dense + CLR	$128(\times 2)$		–	
ABBA Dense + CLR	$64(\times 2)$		–	
ABBA Dense + CLR	$32(\times 2)$		–	
Dense	3		2	

Table SM3

ABBA Dense and ABBA Conv architecture details for MNIST and FMNIST datasets. For convolutional layers the stride is set to 1.

Layer type	MNIST/FMNIST
Input	$28 \times 28 \times 1$
Conv2D	$28 \times 28 \times 32$
ABBA Conv2D + CLR	$28 \times 28 \times 16$
ABBA Conv2D + CLR	$28 \times 28 \times 16$
Conv2D	$28 \times 28 \times 1$
Dense	256
ABBA Dense + CLR	128
ABBA Dense + CLR	64
Dense	10

Layer type	MNIST	FMNIST
Input	784	784
Dense	256	256
ABBA Dense + CLR	128	128
ABBA Dense + CLR	64	64
ABBA Dense + CLR	–	32
Dense	10	10

Table SM4

Training hyperparameters.

Dataset	Optimizer	No. Epochs	Learning rate	Batch size
MNIST	projected ADAM	150	10^{-3}	1024
FMNIST	projected ADAM	200	10^{-3}	1024
RPS	projected ADAM	250	10^{-4}	64
CelebA	projected ADAM	100	10^{-4}	128

randomly select the same number of examples from the *non-bald* class, in order to avoid class imbalance. Additional information regarding the optimization parameters used during training is provided in Table SM4.

SM10. Adversarial examples. For all datasets, adversarial examples created by using DDN attack are displayed in Figures SM2, SM3, SM4, and SM5. We generated adversarial samples using untargeted DDN attacks, with a budget of 300 iterations and initial parameters as proposed by the authors. We did not limit the maximum perturbation ϵ, in order to find the minimum one allowing us to fool the model. It can be easily seen that for DeelLip and ABBA networks the required perturbations for misclassification are higher. In particular, we observe that the perturbations needed to fool ABBA networks lead to severe artifacts in the images.

SM11. Training time. We first compared the average time/epoch for training a standard network and its ABBA equivalent. Table SM5 reports the average seconds per epoch for both cases, for different feed-forward architectures. On average, training an ABBA neural network for 200 epochs on MNIST introduces less than 10% additional training time.

The projection is a costly step, and it is the main contributor to the training overhead. A comparison of the training time (per-batch) with (*green line*) and without (*dotted green line*) projection is featured in Figures SM6a and SM6b for architectures with an increasing number of fully connected and convolutional layers, respectively. The average deviation from the

Figure SM2. *Adversarial examples with DDN attack for Conv-Dense models, on MNIST dataset.* ℓ_2 *perturbation magnitude is given in the top-left corner.*

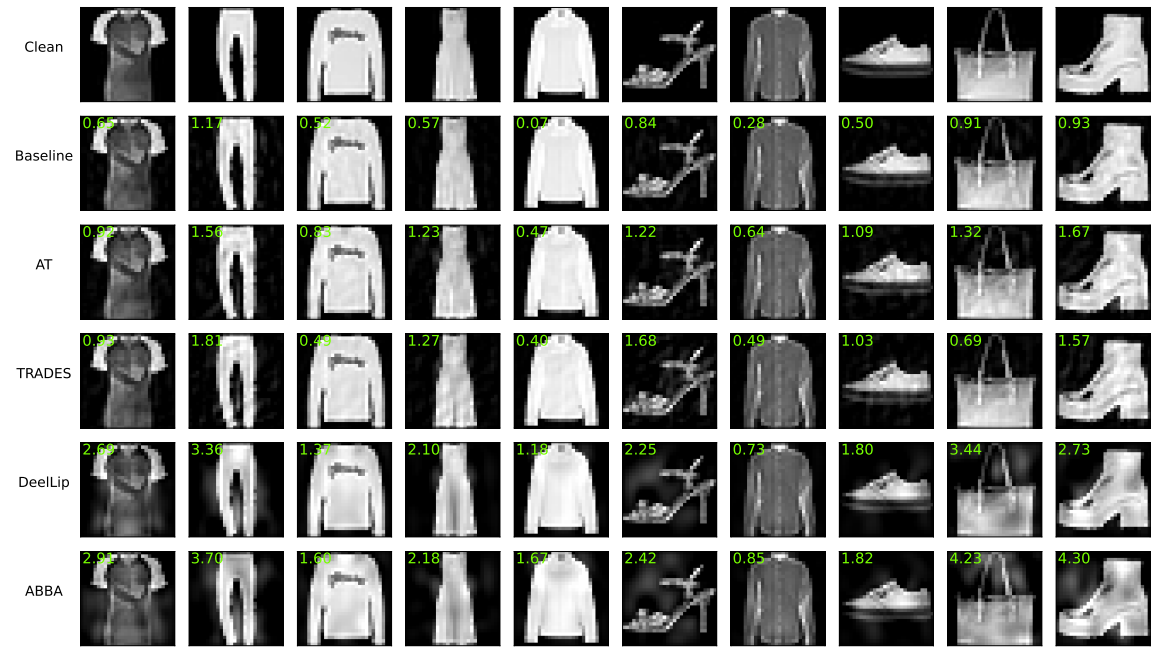


Figure SM3. *Adversarial examples with DDN attack for Conv-Dense models, on FMNIST dataset.* ℓ_2 *perturbation magnitude is given in the top-left corner.*

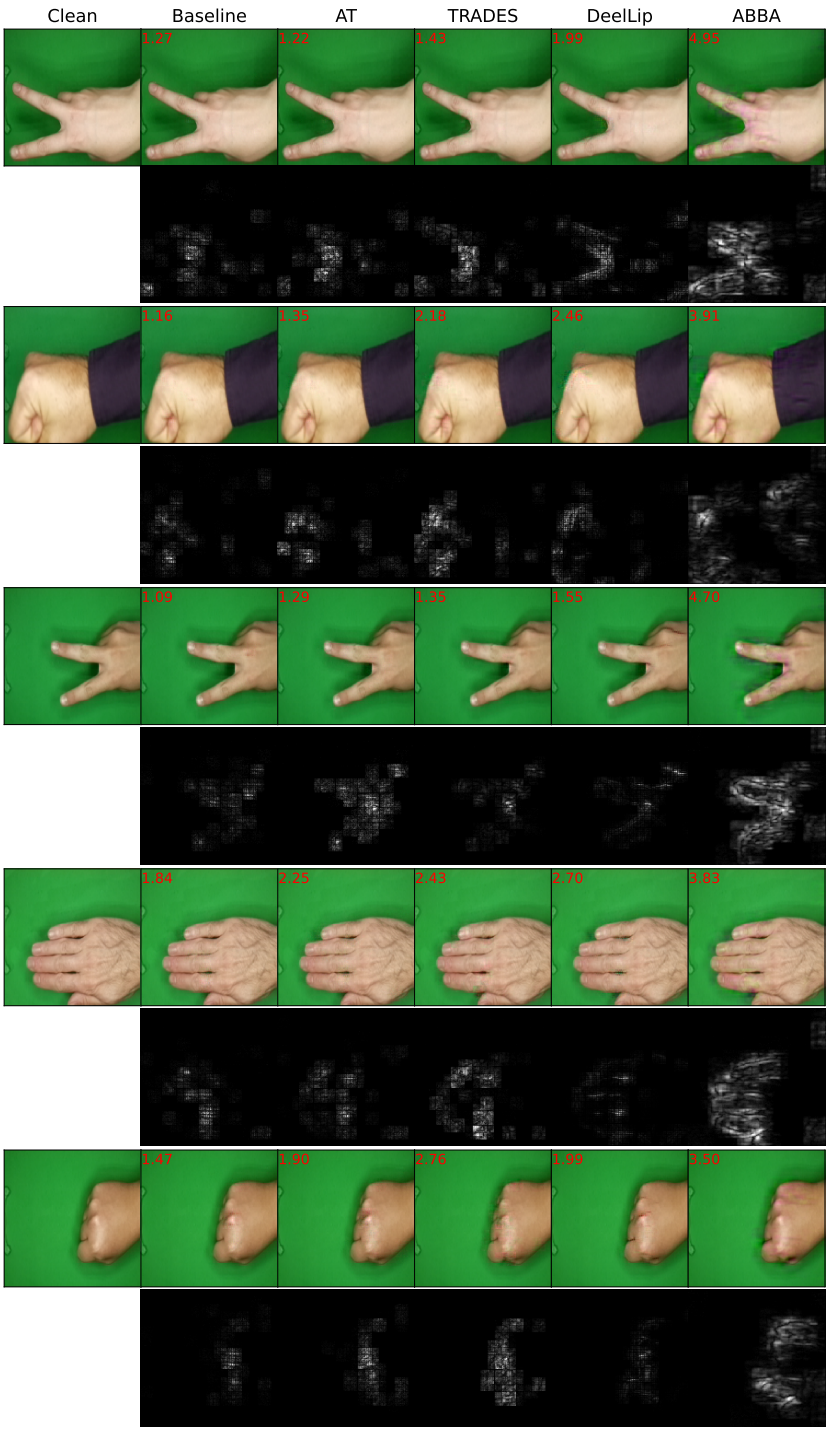


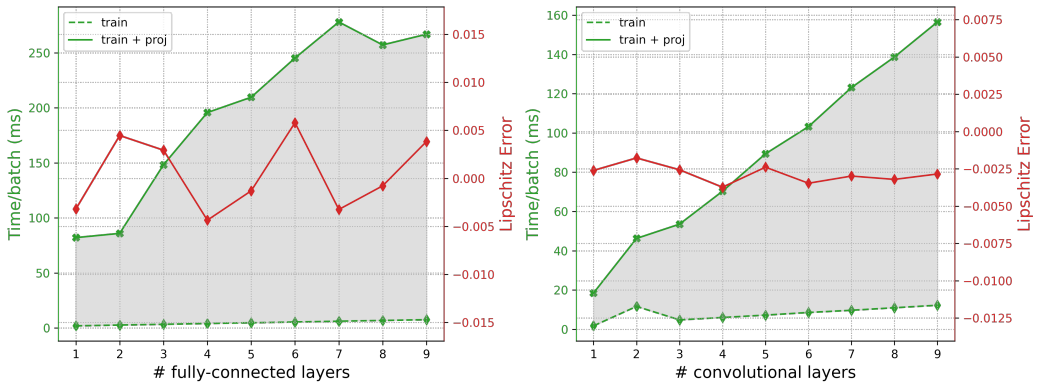
Figure SM4. *Adversarial examples generated with DDN, on RPS dataset. For each example: first row – adversarial images; second row – pixel differences between adversarial and clean sample.*

Figure SM5. *Adversarial examples with DeepFool attack for CelebA. ℓ_2 perturbation magnitude is given in the top-left corner.*

Table SM5

Comparison of per-epoch training times for various Standard and ABBA architectures, on MNIST. XCYF corresponds to an architecture with X Convolution layers, followed by Y fully-connected layers.

Model		Architecture					
		2C2F	2C3F	3C2F	4C2F	4C3F	5C1F
Standard	Acc [%]	95.28	95.80	99.18	99.30	99.26	99.10
	Sec./Epoch	4.25	4.29	4.31	4.28	4.37	4.31
	Size (MB)	0.09	0.14	0.39	0.53	0.59	0.97
ABBA	Acc [%]	95.54	95.34	98.62	99.12	99.14	98.72
	Sec./Epoch	4.52	4.61	4.67	4.68	4.79	4.74
	Size (MB)	0.18	0.28	0.78	1.06	1.18	1.94



(a) (b)

Figure SM6. *Computation time for the projection step for a variable-length sequence of ABBA* SM6a *fully-connected and* SM6b *convolutional layers. All projections wer computed with a number of* 10 *iterations, and the results were averaged over* 50 *independent simulations.*

imposed global bound (*red*), which was set to 1 in all cases, is also reported. This shows that we are able to maintain the imposed bounds, given the same number of iterations, irrespective of the network depth.

REFERENCES

[SM1] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert spaces*, 2nd ed., corrected printing. New York: Springer, (2019).

[SM2] P. L. Combettes and J.-C. Pesquet, *Proximal splitting methods in signal processing*, Fixed-Point Algorithms for Inverse Problems in Science and Engineering, (2011), pp. 185–212.

[SM3] P. L. Combettes and J.-C. Pesquet, *Lipschitz certificates for layered network structures driven by averaged activation operators*, J. Math. Data Sci., 2 (2020), pp. 529–557.

[SM4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition*, Proc. of the IEEE, 86 (1998), pp. 2278–2324.

[SM5] J. R. Silvester, *Determinants of block matrices*, The Mathematical Gazette, 84 (2000), pp. 460–467.

[SM6] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*, Prentice Hall, (1993).